Mini Mars Rover

Submitted to

(Elizabeth Freije and Andrew McNeely)
Engineering Technology Department

by

Nafa Alanzi
Chad Tan
Dae'Shaun Walden

December 11, 2020

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The customer, Andrew McNeely, would like us to construct and design a miniature sized version of the existing Mars Rover robot. The robot will be controlled through an Android Application that we have designed, that will control the motor movements and command the robot to collect five data points from the environment. What we are given that is out of our scope for operation is an existing robotics kit that we will grab components from, the battery, frame, wheels, and motors. Our In-Scope of operation is to design a buck converter, power supply, and a transistor circuit that will transfer a low voltage output from a higher voltage input, a battery. We also designed a board layout for the motor control and designed a code for the Android Application and the microcontroller.

REVISION HISTORY

Version	Date	Revised by	Description
1.0	28 November 2020	Dae'Shaun Walden	Initial version
1.1	5 December 2020	Dae'Shaun Walden	Revised in proper format
1.5	8 December 2020	Dae'Shaun Walden	Added to the notes section, revised the project poster, add more parts to the appendices section
2.0	10 December 2020	Nafa, Chad, Dae'Shaun	Work on the format , add Weekly reports
2.1	11 December 2020	Nafa, Chad, Dae'Shaun	Updated all sections

INTRODUCTION/SCOPE/MARKETING REQUIREMENTS

Multiple rover style robots have been sent to Mars to collect data for research as early as 1997, which today is still being improved upon with the fast advancements in technology. Soon I am sure more rover style robots will be launched to Mars for more research which can be used to observe if the planet is suitable for humans or signs of any life. We have designed and are constructing a smaller but similar version of the Mars rover from NASA. Within this paper, the customer describes how they wanted the rover to operate and function. We as the engineers' job is to design all the necessary circuits and programs to build the rover to meet the specifications. Since our Senior Design Project Phase II is based off the customer's specifications, phase I, and many changes to our project we must be sure that all our previous planning and designing along with the customer's request is met when building the robot with implementing the different components we designed and calculate to enable the rover to operate. Also leaving time to debug and adjust when everything is connected so it will operate exactly how we and the customer want it to.

The project is designing and building a robot that can be commanded wirelessly through a phone and collect data from the environment that is sent back to the phone. The customer for this project is Andrew McNeely, a faculty member at the Department of Engineering Technology at the campus of IUPUI. His reason for this project is to show the capabilities of engineering technology students, and to show off a robot to the robotics club.

The scope of this project is to architect, design, and create the Mini-Mars Rover.

- The coded software will allow the robot to operate based on our commands
- Sensors will allow the Microcontroller to collect the data from the environment.
- A power supply will allow the motors, transistor circuit and microcontroller on the robot to operate.
- The robot frame will house all the components

Identification

The Mini-Mars Rover should run for one or more hours, be controlled through a phone by wirelessly connectivity, and send back data from the environment (Temperature, Altitude, Pressure, Humidity, and Position) to the phone. The battery of the robot should be charged through plug in.

System Overview

Source	Specification	Justification	Priority
Customer	Power Supply that can run 1 hour		Runtime
Customer	Collect 5 Environmental Data (Pressure, Humidity, Altitude, Proximity, Temperature)		Data Collection
Customer	Use robot frame, 2 motors from the faculty advisor and wheels from VEX V5 Robot Kit		Robot Construction

Document Overview

Source	Requirement	Priority
System Documentation	● Component Listing and Specs ● Charts ● Schematics	High
User Manual	Easy to read manual of how to operate the robot using the phone & maintenance	High

In Scope

- Wiring components
- Coding software and commands
- Construct the robot
- Design power supply
- Design transistor circuit
- Buying electrical components
- Proximity sensors in the front
- One-speed motor
- Distance travel
- Onboard power
- Five Sensors
  - Altitude
  - Temperature
  - Pressure
  - Humidity
  - Proximity (Position)
  -

Out of Scope

- Buy a phone
- Testing area
- Selecting the robot kit (provided by the university)
- A design of control board
- Speed control and Position detection

SPECIFICATION REQUIREMENTS

The customer has enlisted several requirements, restrictions, and changes for the project that we must be sure to incorporate into our project to mirror the idea and big picture of the customer. The Mini-Mars Rover must operate including executing commands and running within at least a one-hour span. Previously during non-operation periods, standby, or operation periods, the robot will be charged through solar panels but at the end of our Phase I the customer and faculty advisors recommended that we take out this section of the project and use the battery that comes with the Vex V5 kit. The commands for the rover will be communicated through Bluetooth connectivity, with the use of an Android device and Bluetooth module. The commands sent to the robot will be sent as a packet and operated depending on the order of the commands sent. This idea means that the rover is not designed for live feedback until the packet of commands has been sent and received. The rover will receive two forms of commands including directional movement and the collection of data. For directional commanding, the robot will only go forward 3 feet or turn left or right 30 degrees per command. The rover will execute the commands one at a time in the order entered by the user. A proximity sensor will be located at the front of the robot that will allow the robot to move 3 feet forward only if the rover is not being blocked by an item or an item is within the 3 feet distance. For the collection data, the rover will send back the necessary environmental data including, humidity, temperature, pressure, altitude, and proximity (position). This will be sent back to the android device. The restrictions that the customer provided for the project are:

(1) No breadboard can be on the robot.

Previously the customer has provided several pieces of equipment that we can use for the Mini Mars Rover project, which all come from the robotics kit, VEX V5 Robot Kit.

(1) Two motors

(2) Robot frame

(3) Robot wheels

(4) Wireless phone (Android)

Now with all the different changes that we made to our project, we notice we cannot use the motors and the frame from the VEX V5 Robot Kit. We were provided a different frame and motors from a faculty advisor in conjunction with parts from the VEX V5 Robot Kit.

(1) Robot wheels
(2) VEX robot battery and cable
(3) Wireless phone (Android)
(4) Motors from faculty advisor
(5) Frame from faculty advisor

Finally, the customer has given us the testing conditions that the robot will be allowed to operate within:

(1) Temperature: 40 - 85 degrees Fahrenheit

(2) Weather Conditions: Clear skies to light rain

For the environmental sensors:

(1) Ensure the sensors are receiving sending data from the command packet back to the Android phone.

For the microcontroller, power supply, and transistor circuit:

(1) Ensure power from the battery, buck converter, and microcontroller can power the transistor motor control circuit to enable the motors to move.

For the frame:

(1) Ensure it can hold the multiple boards of the power supply, microcontroller, and transistor circuit

(2) Ensure it can hold the battery of the rover and can have wheels mounted

Previously, we had to calculate the size of battery that we need to operate within our allocated one hours or more for the customer's approval but now since we are using the battery from the robotics kit which was a major change in the project along with eliminating the solar panel aspect of the rover. To protect the microcontroller and transistor circuit from being overheated from the battery, we incorporated a buck converter into the battery to the transistor motor control circuit and then to the microcontroller. This is the brains of the rover to enable it to move and sense data.

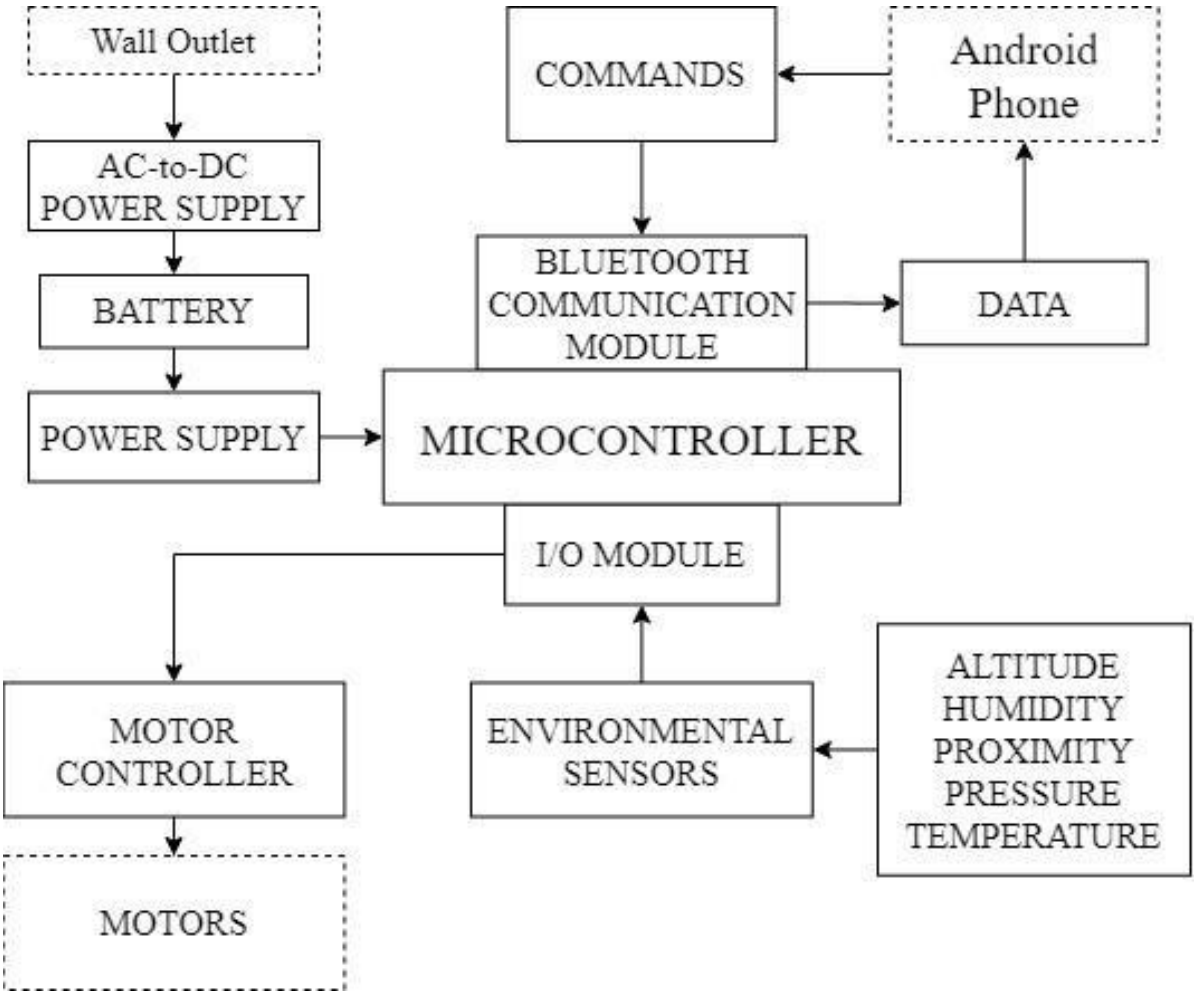
HIGH LEVEL DESIGN

System – Wide Design Decisions

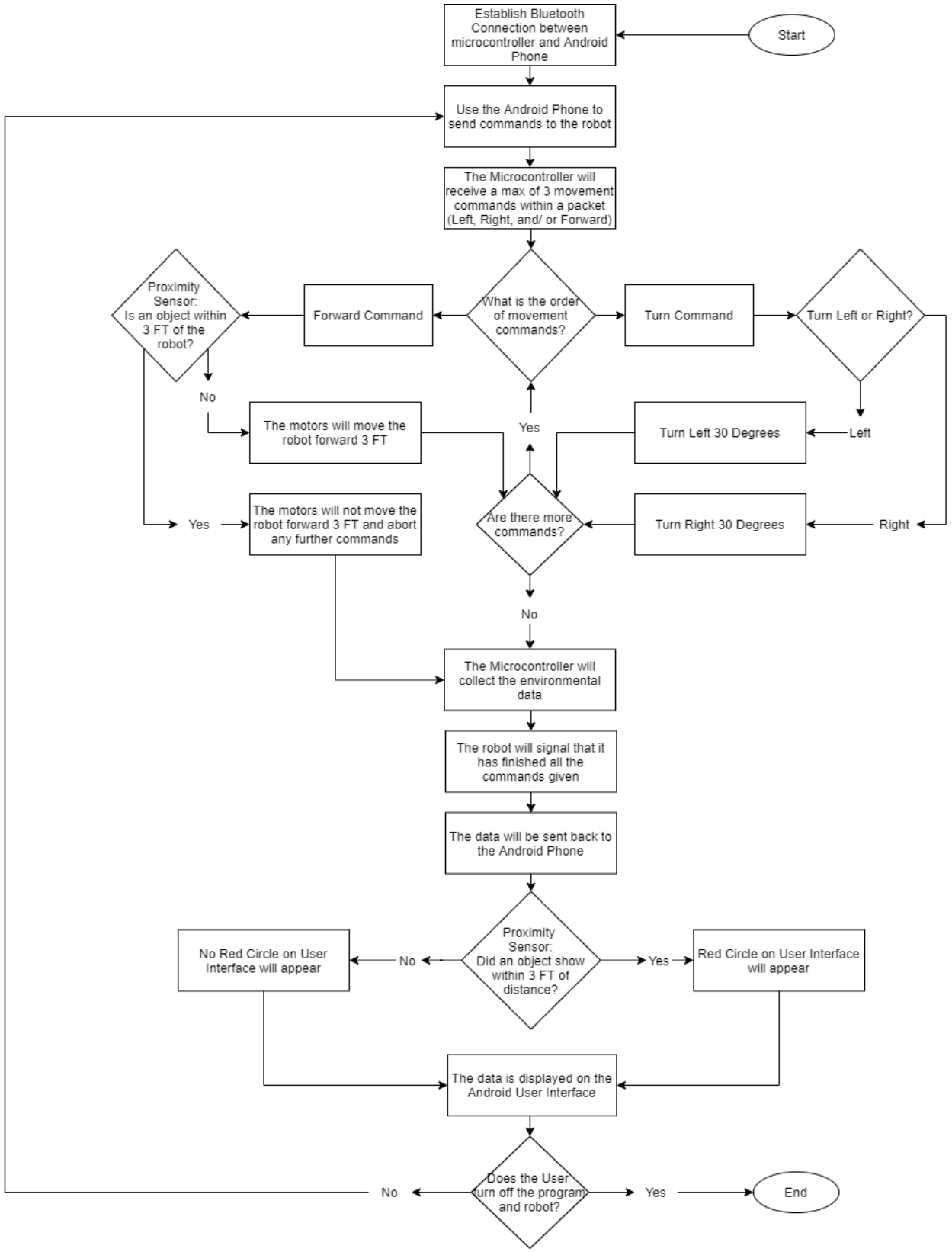
Major Components

- **Battery/ Power Supply:** This will provide power to the entire robot, allowing it to operate. This will be designed by us to operate within the specified 1 hour or more operation time requested by the customer.
- **Microcontroller:** This will be the brains of operating the robot and communication with the Android phone. Once programmed, the microcontroller will communicate with the phone through Bluetooth, operate the commands received by the phone if applicable, and return data received by the sensors.
- **Sensors:** Will collect data from the environment to be sent back to the operator's phone to be read. These sensors will collect the altitude, humidity, pressure, position, and temperature of the given area.

System Architectural Design



[Figure 1: Hardware System Architecture, System Components]



[Figure 2: Software System Architecture, Concept of Execution]

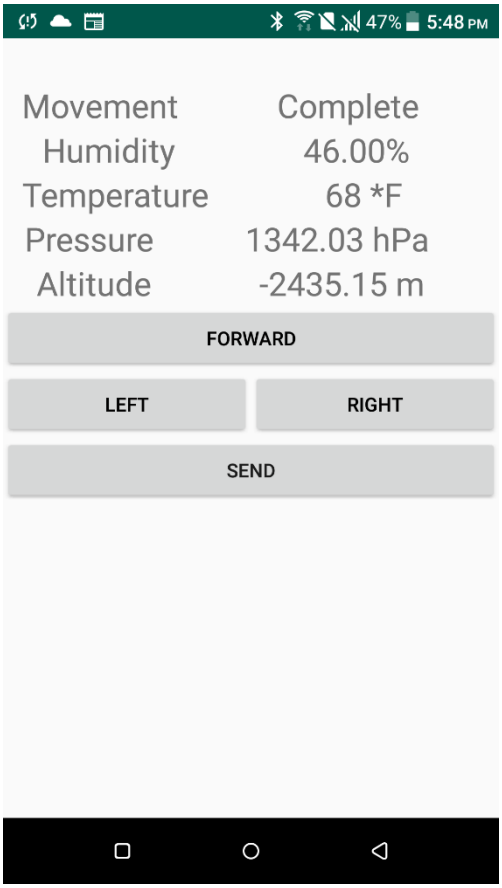
System Components

- **Android Phone:** This device will be provided; however, to briefly explain that this device is the controller to operate the robot, send commands to the robot, and receive data from the robot. We are responsible to make a program that can send and receive data from the robot. We will provide the code in a zipped program file that will include full operation instructions on how to install it to an Android phone and connect to the robot via Bluetooth
- **Battery:** The battery is the main power source that will be providing voltage to the microcontroller without the need for an outlet. This is a necessary component of the robot because the robot will be a mobile device that is constantly moving in the environment and will be inhibited and inefficient to traverse. Within this component will be a buck converter between the battery, transistor circuit and microcontroller so as not to break the microcontroller with more voltage than it needs also for the microcontroller can use the transistor circuit as a digital switch
- **Microcontroller:** This device is the brains of the entire robot. Once the Bluetooth module is connected to the device, it will allow communication between the robot and the phone.
- **Transistor Circuit/Motor Controller:** This will allow the microcontroller to send the signal to the motors when needed to move. Since the microcontroller is not capable of sending enough voltage to the motors.
- **Environmental Sensors:** These devices will collect the data from the environment which will be read by the microcontroller.

Concept of Execution

- **Android Phone Send:** We will use software that is designed to integrate with the microcontroller through a Bluetooth module to act as a controller.
- **Sensors collecting environmental data:** The sensors will automatically collect the data from the environment and will send back the data to the phone on User's command. The proximity sensor will also be utilized for the forward movement of the robot.
- **Received Movement Commands:** Rather than the robot operating under live feedback after receiving a single movement command, we have been tasked by the customer to send a packet of commands to the robot to operate in order of the commands, like the Mars Rover.
- **Motor Operations:** The motors will cause the robot to only operate in three directions (Left, Right, or Forward). When turning, it will one turn 30° from its position in a left or right direction. When moving forward, it will only operate this command when there is nothing obstructing the robot to move forward 3 feet; otherwise, it will operate this command.
- **Android Phone Receive:** The Android phone will receive the environmental data and be presented within the user interface after the robot has finished it's packet of commands.

Interface Design



[Figure 3: User Interface]

- **Inputs:** The left, right, and forward keys are movement commands that the robot will receive, making the motors operate to perform these movement commands. ○ **Forward** ○ **Left/ Right**
- **Outputs:**
    - **Altitude, Humidity, Pressure, and Temperature:** Will display numbers and data collected from the environment with corresponding units next to them.
    - **The green circle**: This is a send command after selecting all the sensors and movement the user wants.
    - **The red circle:** This circle will emit when the proximity sensor detects that an object is obstructing the robot to move or one the sensors did not work. A zero value will be displayed from the sensor that has the error.

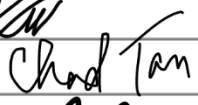
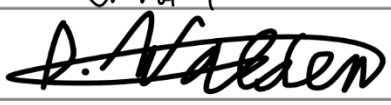
This design was chosen as it displays all the required information that the customer has requested on one screen rather than shifting through several tabs. This also accommodates individuals who decide to operate the phone one-handed without covering the environmental data being displayed.

Document Approval

This section includes signatures of those that have written, reviewed, or approved this High-Level Design deliverable for the Mini-Mars Rover.

Authors' Signatures:

Your signature indicates that this document describes the Requirements of the Mini-Mars Rover and that this deliverable meets IEE standards for documentation.

Name	Signature	Title/Department	Date
Alanzi, Nafa		Author	03/13/20
Tan, Chad Q.		Author	03/13/20
Walden, Dae'shaun		Author	03/13/20

Approvers' Signatures:

Your signature signifies that you agree with the high-level design presented in this document, and that it has been reviewed by appropriate personnel to ensure compliance with company and/ or regulatory policies.

Name	Signature	Title/Department	Date
McNeely, Andrew		Project Sponsor	
		Process Owner	
		Project Manager	
		Application Architect	
		Quality Manager	

LOW LEVEL DESIGN

System Overview

Source	Specification	Justification	Priority
Customer	Power Supply that can run 1 hour		Runtime
Customer	Batteries are charged by plug in.		Battery Charger
Customer	Collect 5 Environmental Data (Pressure, Humidity, Altitude, Proximity, Temperature)		Data Collection
Customer	Use robot frame, 2 motors from faculty advisor and wheels from VEX V5 Robot Kit		Robot Construction Kit

Document Overview

Source	Requirement	Priority
System Documentation	• Component Listing and Specs • Charts • Schematics	High
User Manual	Easy to read manual of how to operate the robot using the phone & maintenance	High

Referenced Documents

Table 5 – Standards and Related Documents

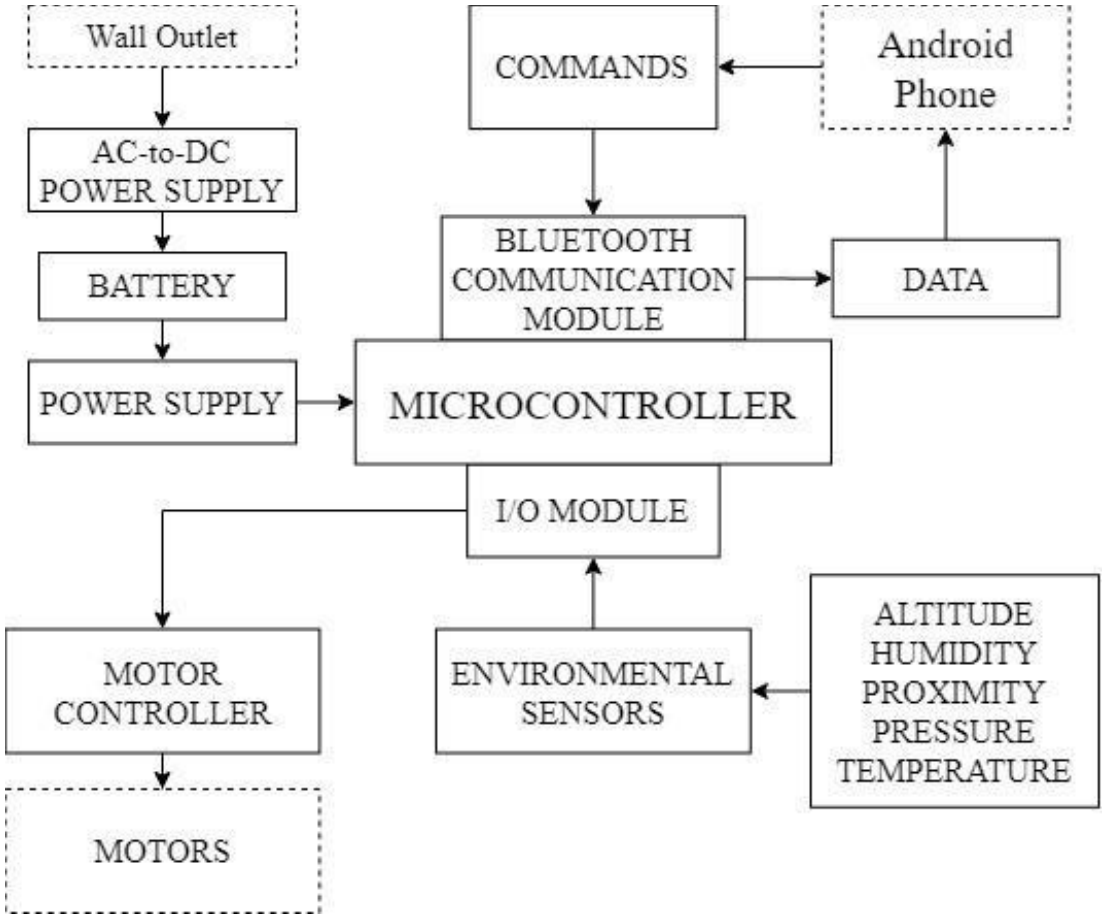
Title	Source	Comment
V5 Vex Robot	vexrobotics	The robot kit
MSP432P401R LaunchPad	Texas Instruments	Microcontroller
LM2575	Texas Instruments	Buck Converter
BMP280	Bosch	Temperature, Pressure, Altitude Sensor
DHT11	Mouser	Humidity, Temperature Sensor
JSN-SR04T	Datasheet	Proximity Sensor

System – Wide Design Decisions

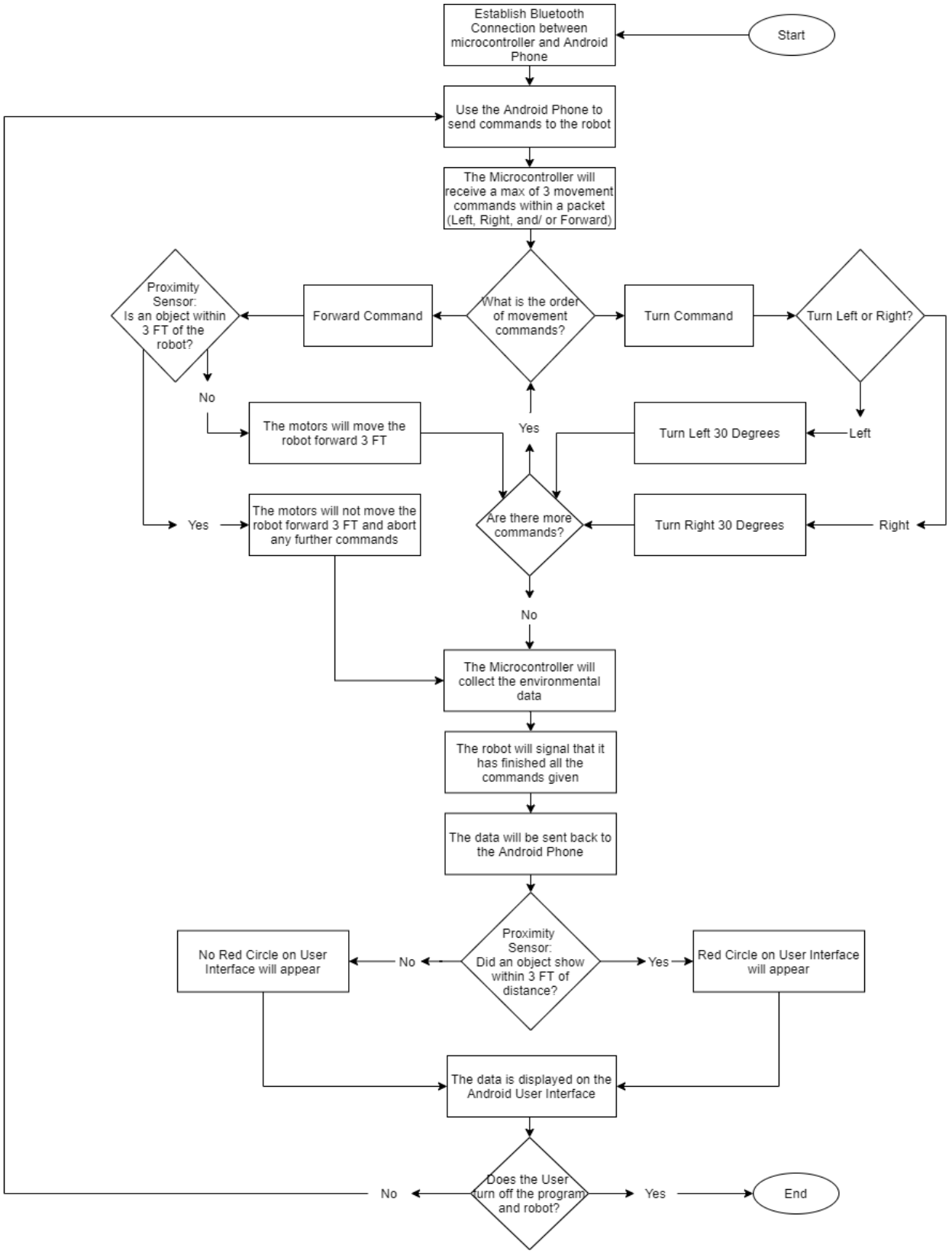
Major Components

- **Battery/ Power Supply:** This will provide power to the entire robot, allowing it to operate. This will be designed by us to operate up to an hour of runtime and will be charged through an outlet (plug).
- **Microcontroller:** This will be the brains of operating the robot and communication with the Android phone. Once programmed, the microcontroller will communicate with the phone through Bluetooth, operate the commands received by the phone if applicable, and return data received by the sensors.
- **Sensors:** Will collect data from the environment to be sent back to the operator's phone to be read. These sensors will collect the altitude, humidity, pressure, position, and temperature of the given area.

System Architectural Design



[Figure 1: Hardware System Architecture, System Components]



[Figure 2: Software System Architecture, Concept of Execution]

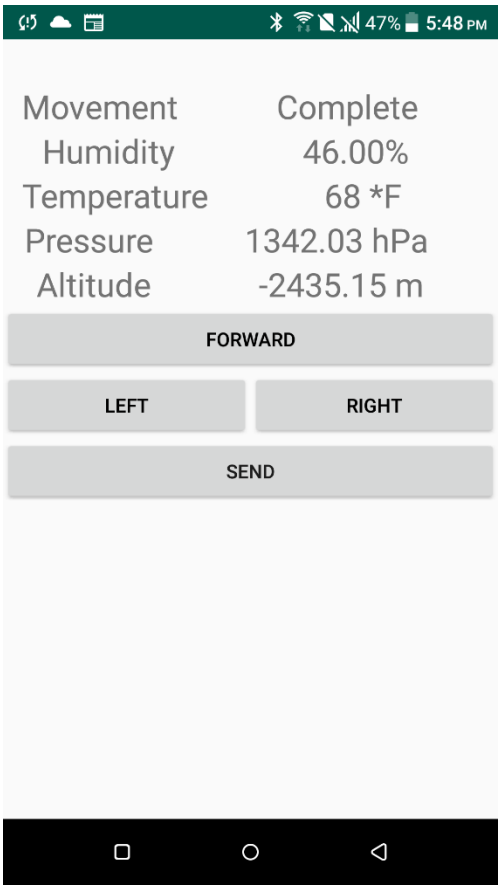
System Components

- **Android Phone:** This device is provided; however, to briefly explain that this device is the controller to operate the robot, send commands to the robot, and receive data from the robot.
  We are responsible to make a program that can send and receive data from the robot.
- **Battery:** The battery is the main power source that will be providing voltage to the power supply then to the microcontroller. This is a necessary component on the robot because the robot will be a mobile device that is constantly moving in the environment. Within this component will be a buck converter between the battery and microcontroller so as not to break the microcontroller with more voltage than it needs.
- **Microcontroller:** This device is the brains of the entire robot. Once the Bluetooth module is connected to the device, it will allow communication between the robot and the phone.
- **Motor Controller:** This will allow the microcontroller to send the signal to the motors when needed to move. Since the microcontroller is not capable of sending enough current to the motors.
- **Environmental Sensors:** These devices will collect the data from the environment which will be read by the microcontroller.
- **Plug-in-Wall:** 120V supplied to the AC-to-DC power supply.

Concept of Execution

- **Android Phone Send:** Will have software that is designed to be used to integrate with the microcontroller through a Bluetooth module to act as a controller.
- **Sensors collecting environmental data:** The sensors will automatically collect the data from the environment and will send back the data to the phone on User's command. The proximity sensor will also be utilized for the forward movement of the robot.
- **Received Movement Commands:** Rather than the robot operating under live feedback after receiving a single movement command, we have been tasked by the customer to send a packet of commands to the robot to operate in order of the commands, similar to the Mars Rover.
- **Motor Operations:** The motors will cause the robot to only operate in three directions (Left, Right, or Forward). When turning, it will one turn 30° from its position in a left or right direction. When moving forward, it will only operate this command when there is nothing obstructing the robot to move forward 3 feet; otherwise, it will operate this command.
- **Android Phone Receive:** The Android phone will receive the environmental data and be presented with a signal of some form that the robot has finished its packet of commands.

Interface Design



[Figure 3: User Interface]

- **Inputs:**
  - The directional arrow keys are movement commands that the robot will receive, making the motors operate to perform these movement commands. Each directional input can only be used once and will be highlighted which commands will be operated. The commands will operate based on the order of which was pressed.
    - **Forward 3 Feet**
    - **Left/ Right 30°** ○ **The green circle (Send)**: This is a send command after selecting all the sensors and movement the user wants.
- **Outputs:**
  - **Altitude, Humidity, Pressure, and Temperature:** Will display numbers and data collected from the environment with corresponding units next to them.
  - **The red circle (Error):** This circle will emit when the proximity sensor detects that an object is obstructing the robot to move or one the sensors did not work. A zero value will be displayed from the sensor that has the error.

This design was chosen as it displays all the required information that the customer has requested on one screen rather than shifting through several tabs. This also accommodates individuals who

decide to operate the phone one-handed without covering the environmental data being displayed.

Detailed Hardware Design

Sub Assembly 1 Schematic & Operation

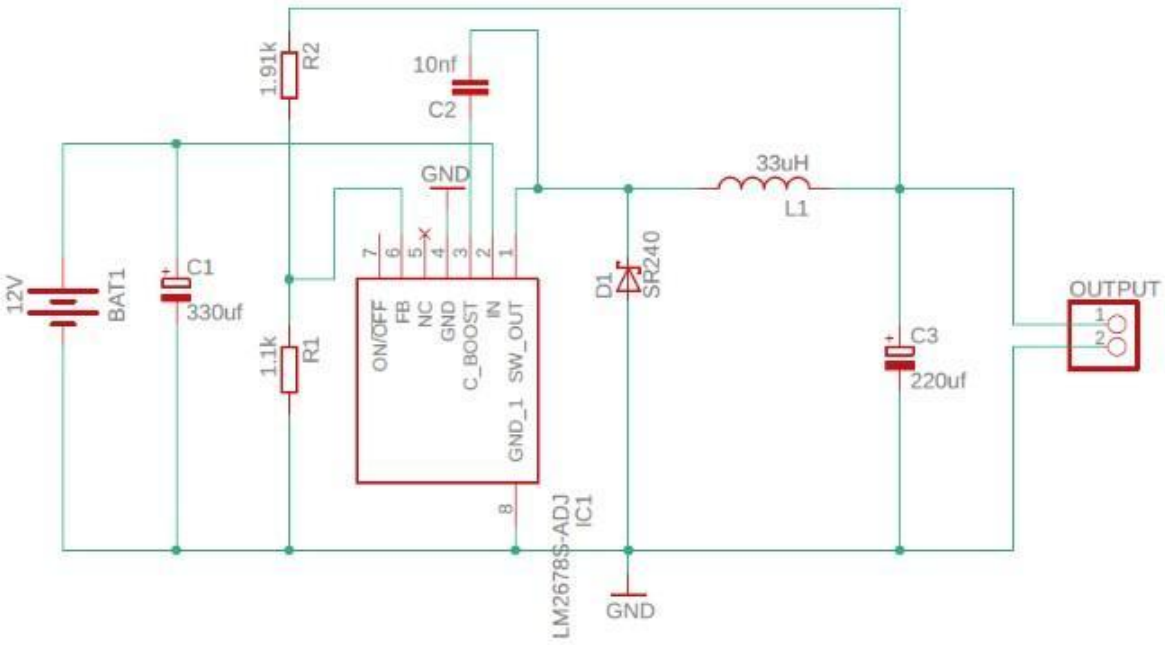
Power supply: Buck Converter Design using LM2678

Buck Converter Design using LM2678

V out = 3.3V

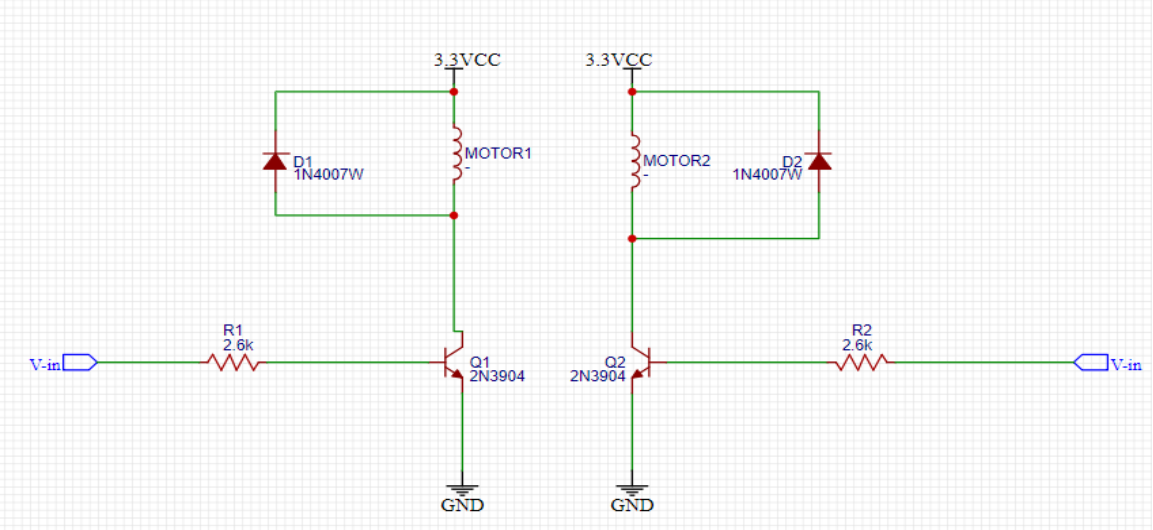
I out = 1.2A

Vin = 12V



Sub Assembly 2 Schematic & Operation

Transistor Circuit/Motor Control: Using a 90Ω resistor



Detailed Software Design

Microcontroller:

Credit to Engr Fahad from Electronic Clinic:

- **Control Arduino over Bluetooth using Android Studio**

Credit to MIKROBLOG:

- **BMP280 barometric pressure sensor example**

Credit to Arbi Abdul Jabbaar:

- **Ultrasonic Sensor HC-SR04**

Credit to ElectricWings:

- **DHT11 Sensor Interfacing with MSP-EXP430**

Main Program

```
Serial.begin(9600);       //Define baud rate for serial communication //

 //BMP280
 // Initialise I2C communication as MASTER
 Wire.begin();
 //BMP280

 //DHT11
 pinMode(40, OUTPUT);
 pinMode(39, OUTPUT);

 // Initialize device.
 dht.begin();
```

```
sensor_t sensor;
dht.temperature().getSensor(&sensor);

dht.humidity().getSensor(&sensor);

// Set delay between sensor readings based on sensor details.
delayMS = sensor.min_delay / 1000;
//DHT11

//HC-SR04
 pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
```

Subroutine 1

```
void loop() {
 //Switch Case
 if (Serial.available()>0)  /* If data is available on serial port */
   {
    char data_received;
    //String input;
    data_received = Serial.read();  /* Data received from bluetooth */

    //
    int temperature;
    //

        //Turn robot left or right for .25 ms
    if (data_received == 13)
    {
     if(stop == 0)
     {
      digitalWrite(40, HIGH);

      delay(250);

      digitalWrite(40, LOW);
     }

     return;

    }

        //Move both motors forward for 2.5 seconds
    else if (data_received == 11)
    {
     total = 0;
```

```
  for(int thisReading = 0; thisReading < numReadings; thisReading++)
  {
   //HC-SRO4 Proximity Sensor
     // Clears the trigPin condition
     digitalWrite(trigPin, LOW);
     delayMicroseconds(2);
     // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
     digitalWrite(trigPin, HIGH);
     delayMicroseconds(10);
     digitalWrite(trigPin, LOW);
     // Reads the echoPin, returns the sound wave travel time in microseconds
     duration = pulseIn(echoPin, HIGH);
     // Calculating the distance
     distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)

     total += distance;

     delay(1);
   //HC-SRO4 Proximity Sensor
  }
  // calculate the average:
     average = total / numReadings;

      //Robot will only move if clear within 3 FT (92)
  if(average >= 92)
  {
   digitalWrite(39, HIGH);
   digitalWrite(40, HIGH);

   delay(2500);

   digitalWrite(40, LOW);
   digitalWrite(39, LOW);
  }
  else
  {
   stop = 1;
  }

}

      //Turn robot left or right for .25 ms
else if (data_received == 12)
{
  if(stop == 0)
  {
   digitalWrite(39, HIGH);

   delay(250);

   digitalWrite(39, LOW);
  }

   return;
```

```
}
else if (data_received == 14)
{
 //DHT11 Read Data  (Temperature and Humidity)
 // Delay between measurements.
 delay(delayMS);
 // Get temperature event and print its value.
 sensors_event_t event;
 dht.temperature().getEvent(&event);
 if (isnan(event.temperature)) {
  Serial.println(F("Error reading temperature!"));
 }
 else {

  temperature = (event.temperature * 1.8) + 32;
  Serial.print(temperature);
  Serial.println(F(" *F"));
 }
 // Get humidity event and print its value.
 dht.humidity().getEvent(&event);
 if (isnan(event.relative_humidity)) {
  Serial.println(F("Error reading humidity!"));
 }
 else {
  //Serial.print(F("Humidity: "));
  Serial.print(event.relative_humidity);
  Serial.println(F("%"));
  //Serial.println(" ");
     }

 delay(100);
 //DHT11 Read Data (Temperature and Humidity)

 //BMP280 Read Pressure and Temperature
 unsigned int b1[24];
 unsigned int data[8];
 for (int i = 0; i < 24; i++)
 {
  // Start I2C Transmission
  Wire.beginTransmission(Addr);
  // Select data register
  Wire.write((136 + i));
  // Stop I2C Transmission
  Wire.endTransmission();

  // Request 1 byte of data
  Wire.requestFrom(Addr, 1);

  // Read 1 byte of data
  if (Wire.available() == 1)
  {
   b1[i] = Wire.read();
  }
 }
```

```
// Convert the data
// temp coefficients
unsigned int dig_T1 = (b1[0] & 0xFF) + ((b1[1] & 0xFF) * 256);
int dig_T2 = b1[2] + (b1[3] * 256);
int dig_T3 = b1[4] + (b1[5] * 256);

// pressure coefficients
unsigned int dig_P1 = (b1[6] & 0xFF) + ((b1[7] & 0xFF) * 256);
int dig_P2 = b1[8] + (b1[9] * 256);
int dig_P3 = b1[10] + (b1[11] * 256);
int dig_P4 = b1[12] + (b1[13] * 256);
int dig_P5 = b1[14] + (b1[15] * 256);
int dig_P6 = b1[16] + (b1[17] * 256);
int dig_P7 = b1[18] + (b1[19] * 256);
int dig_P8 = b1[20] + (b1[21] * 256);
int dig_P9 = b1[22] + (b1[23] * 256);

// Start I2C Transmission
Wire.beginTransmission(Addr);
// Select control measurement register
Wire.write(0xF4);
// Normal mode, temp and pressure over sampling rate = 1
Wire.write(0x27);
// Stop I2C Transmission
Wire.endTransmission();

// Start I2C Transmission
Wire.beginTransmission(Addr);
// Select config register
Wire.write(0xF5);
// Stand_by time = 1000ms
Wire.write(0xA0);
// Stop I2C Transmission
Wire.endTransmission();

for (int i = 0; i < 8; i++)
{
 // Start I2C Transmission
 Wire.beginTransmission(Addr);
 // Select data register
 Wire.write((247 + i));
 // Stop I2C Transmission
 Wire.endTransmission();

 // Request 1 byte of data
 Wire.requestFrom(Addr, 1);

 // Read 1 byte of data
 if (Wire.available() == 1)
 {
  data[i] = Wire.read();
 }
}

// Convert pressure and temperature data to 19-bits
```

```
      long adc_p = (((long)(data[0] & 0xFF) * 65536) + ((long)(data[1] & 0xFF) * 256) +
(long)(data[2] & 0xF0)) / 16;
      long adc_t = (((long)(data[3] & 0xFF) * 65536) + ((long)(data[4] & 0xFF) * 256) +
(long)(data[5] & 0xF0)) / 16;

      // Temperature offset calculations
      double var1 = (((double)adc_t) / 16384.0 - ((double)dig_T1) / 1024.0) * ((double)dig_T2);
      double var2 = ((((double)adc_t) / 131072.0 - ((double)dig_T1) / 8192.0) *
                (((double)adc_t) / 131072.0 - ((double)dig_T1) / 8192.0)) * ((double)dig_T3);
      double t_fine = (long)(var1 + var2);
      double cTemp = (var1 + var2) / 5120.0;
      double fTemp = cTemp * 1.8 + 32;

      // Pressure offset calculations
      var1 = ((double)t_fine / 2.0) - 64000.0;
      var2 = var1 * var1 * ((double)dig_P6) / 32768.0;
      var2 = var2 + var1 * ((double)dig_P5) * 2.0;
      var2 = (var2 / 4.0) + (((double)dig_P4) * 65536.0);
      var1 = (((double) dig_P3) * var1 * var1 / 524288.0 + ((double) dig_P2) * var1) / 524288.0;
      var1 = (1.0 + var1 / 32768.0) * ((double)dig_P1);
      double p = 1048576.0 - (double)adc_p;
      p = (p - (var2 / 4096.0)) * 6250.0 / var1;
      var1 = ((double) dig_P9) * p * p / 2147483648.0;
      var2 = p * ((double) dig_P8) / 32768.0;
      double pressure = (p + (var1 + var2 + ((double)dig_P7)) / 16.0) / 100;

         //Calculate Altitude based on pressure
      double alt1 = (pressure/1013.25);
      double alt2 = pow(alt1, (1/ 5.255));
      double alt3 = 44330 * (1-alt2);

      Serial.print(pressure);
      Serial.println(" hPa");
      Serial.print(alt3);
      Serial.println(" m");

      delay(100);
         //BMP280 Read Pressure and Temperature


      //Movement
      Serial.println("Complete");
      if(stop == 0)
      {
        Serial.print("No Movement Errors");
      }
      else
      {
        Serial.print("Movement Obstructed");
      }

      stop = 0;
    }
  }
}
```

Android Phone:

Credit to Engr Fahad from Electronic Clinic:

- **Android App for Arduino Sensor Monitoring over Bluetooth**
- **Control Arduino over Bluetooth using Android Studio**

Manifest:AndroidManifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.anysensormonitoring">

   <uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

   <application
       android:allowBackup="true"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:roundIcon="@mipmap/ic_launcher_round"
       android:supportsRtl="true"
       android:theme="@style/AppTheme">
       <activity android:name=".MonitoringScreen" />
       <activity android:name=".MainActivity">
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />

               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
   </application>

</manifest>
```

Java: MainActivity:

```
package com.example.anysensormonitoring;

import android.app.Activity;
//import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.graphics.Color;
import android.os.AsyncTask;
import android.preference.PreferenceManager;
//import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;


public class MainActivity extends Activity {
    private Button search;
    private Button connect;
    private ListView listView;
    private BluetoothAdapter mBTAdapter;
    private static final int BT_ENABLE_REQUEST = 10; // This is the code we
use for BT Enable
    private static final int SETTINGS = 20;
    private UUID mDeviceUUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
    private int mBufferSize = 50000; //Default
    public static final String DEVICE_EXTRA =
"com.example.anysensormonitoring.SOCKET";
    public static final String DEVICE_UUID =
"com.example.anysensormonitoring.uuid";
    private static final String DEVICE_LIST =
"com.example.anysensormonitoring.devicelist";
    private static final String DEVICE_LIST_SELECTED =
"com.example.anysensormonitoring.devicelistselected";
    public static final String BUFFER_SIZE =
"com.example.anysensormonitoring.buffersize";
    private static final String TAG = "BlueTest5-MainActivity";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



        search = (Button) findViewById(R.id.search);
        connect = (Button) findViewById(R.id.connect);

        listView = (ListView) findViewById(R.id.listview);

        if (savedInstanceState != null) {
            ArrayList<BluetoothDevice> list =
savedInstanceState.getParcelableArrayList(DEVICE_LIST);
            if (list != null) {
                initList(list);
                MyAdapter adapter = (MyAdapter) listView.getAdapter();
                int selectedIndex =
savedInstanceState.getInt(DEVICE_LIST_SELECTED);
```

```
                    if (selectedIndex != -1) {
                        adapter.setSelectedIndex(selectedIndex);
                        connect.setEnabled(true);
                    }
                } else {
                    initList(new ArrayList<BluetoothDevice>());
                }

            } else {
                initList(new ArrayList<BluetoothDevice>());
            }
            search.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View arg0) {
                    mBTAdapter = BluetoothAdapter.getDefaultAdapter();

                    if (mBTAdapter == null) {
                        Toast.makeText(getApplicationContext(), "Bluetooth not
found", Toast.LENGTH_SHORT).show();
                    } else if (!mBTAdapter.isEnabled()) {
                        Intent enableBT = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                        startActivityForResult(enableBT, BT_ENABLE_REQUEST);
                    } else {
                        new SearchDevices().execute();
                    }
                }
            });

            connect.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View arg0) {
                    BluetoothDevice device = ((MyAdapter)
(listView.getAdapter())).getSelectedItem();
                    Intent intent = new Intent(getApplicationContext(),
MonitoringScreen.class);
                    intent.putExtra(DEVICE_EXTRA, device);
                    intent.putExtra(DEVICE_UUID, mDeviceUUID.toString());
                    intent.putExtra(BUFFER_SIZE, mBufferSize);
                    startActivity(intent);
                }
            });



        }

        protected void onPause() {
      // TODO Auto-generated method stub
            super.onPause();
        }

        @Override
        protected void onStop() {
      // TODO Auto-generated method stub
            super.onStop();
        }

        @Override
```

```
        protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
            switch (requestCode) {
                case BT_ENABLE_REQUEST:
                    if (resultCode == RESULT_OK) {
                        msg("Bluetooth Enabled successfully");
                        new SearchDevices().execute();
                    } else {
                        msg("Bluetooth couldn't be enabled");
                    }

                    break;
                case SETTINGS: //If the settings have been updated
                    SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
                    String uuid = prefs.getString("prefUuid", "Null");
                    mDeviceUUID = UUID.fromString(uuid);
                    Log.d(TAG, "UUID: " + uuid);
                    String bufSize = prefs.getString("prefTextBuffer", "Null");
                    mBufferSize = Integer.parseInt(bufSize);

                    String orientation = prefs.getString("prefOrientation",
"Null");
                    Log.d(TAG, "Orientation: " + orientation);
                    if (orientation.equals("Landscape")) {

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                    } else if (orientation.equals("Portrait")) {

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                    } else if (orientation.equals("Auto")) {

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
                    }
                    break;
                default:
                    break;
            }
            super.onActivityResult(requestCode, resultCode, data);
        }

        /**
         * Quick way to call the Toast
         * @param str
         */
        private void msg(String str) {
            Toast.makeText(getApplicationContext(), str,
Toast.LENGTH_SHORT).show();
        }

        /**
         * Initialize the List adapter
         * @param objects
         */
        private void initList(List<BluetoothDevice> objects) {
            final MyAdapter adapter = new MyAdapter(getApplicationContext(),
R.layout.list_item, R.id.lstContent, objects);
            listView.setAdapter(adapter);
            listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
```

```
                @Override
                public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                    adapter.setSelectedIndex(position);
                    connect.setEnabled(true);
                }
            });
        }

        /**
         * Searches for paired devices. Doesn't do a scan! Only devices which
are paired through Settings->Bluetooth
         * will show up with this. I didn't see any need to re-build the wheel
over here
         * @author ryder
         *
         */
        private class SearchDevices extends AsyncTask<Void, Void,
List<BluetoothDevice>> {

            @Override
            protected List<BluetoothDevice> doInBackground(Void... params) {
                Set<BluetoothDevice> pairedDevices =
mBTAdapter.getBondedDevices();
                List<BluetoothDevice> listDevices = new
ArrayList<BluetoothDevice>();
                for (BluetoothDevice device : pairedDevices) {
                    listDevices.add(device);
                }
                return listDevices;

            }

            @Override
            protected void onPostExecute(List<BluetoothDevice> listDevices) {
                super.onPostExecute(listDevices);
                if (listDevices.size() > 0) {
                    MyAdapter adapter = (MyAdapter) listView.getAdapter();
                    adapter.replaceItems(listDevices);
                } else {
                    msg("No paired devices found, please pair your serial BT
device and try again");
                }
            }

        }

        /**
         * Custom adapter to show the current devices in the list. This is a bit
of an overkill for this
         * project, but I figured it would be good learning
         * Most of the code is lifted from somewhere but I can't find the link
anymore
         * @author ryder
         *
         */
        private class MyAdapter extends ArrayAdapter<BluetoothDevice> {
            private int selectedIndex;
            private Context context;
            private int selectedColor = Color.parseColor("#abcdef");
            private List<BluetoothDevice> myList;
```

```
            public MyAdapter(Context ctx, int resource, int textViewResourceId,
List<BluetoothDevice> objects) {
                super(ctx, resource, textViewResourceId, objects);
                context = ctx;
                myList = objects;
                selectedIndex = -1;
            }

            public void setSelectedIndex(int position) {
                selectedIndex = position;
                notifyDataSetChanged();
            }

            public BluetoothDevice getSelectedItem() {
                return myList.get(selectedIndex);
            }

            @Override
            public int getCount() {
                return myList.size();
            }

            @Override
            public BluetoothDevice getItem(int position) {
                return myList.get(position);
            }

            @Override
            public long getItemId(int position) {
                return position;
            }

            private class ViewHolder {
                TextView tv;
            }

            public void replaceItems(List<BluetoothDevice> list) {
                myList = list;
                notifyDataSetChanged();
            }

            public List<BluetoothDevice> getEntireList() {
                return myList;
            }

            @Override
            public View getView(int position, View convertView, ViewGroup
parent) {
                View vi = convertView;
                ViewHolder holder;
                if (convertView == null) {
                    vi =
LayoutInflater.from(context).inflate(R.layout.list_item, null);
                    holder = new ViewHolder();

                    holder.tv = (TextView) vi.findViewById(R.id.lstContent);

                    vi.setTag(holder);
                } else {
                    holder = (ViewHolder) vi.getTag();
```

```
                }

                if (selectedIndex != -1 && position == selectedIndex) {
                    holder.tv.setBackgroundColor(selectedColor);
                } else {
                    holder.tv.setBackgroundColor(Color.WHITE);
                }
                BluetoothDevice device = myList.get(position);
                holder.tv.setText(device.getName() + "\n " +
device.getAddress());

                return vi;
            }

        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
            //getMenuInflater().inflate(R.menu.homescreen, menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
                case R.id.action_settings:
                    Intent intent = new Intent(MainActivity.this,
PreferencesActivity.class);
                    startActivityForResult(intent, SETTINGS);
                    break;
            }
            return super.onOptionsItemSelected(item);
        }
    }
```

Java: MonitoringScreen:

```
    package com.example.anysensormonitoring;

    import android.app.Activity;
    import android.app.ProgressDialog;
    import android.bluetooth.BluetoothAdapter;
    import android.bluetooth.BluetoothDevice;
    import android.bluetooth.BluetoothSocket;
    import android.content.Intent;
    import android.graphics.Color;
    import android.os.AsyncTask;
    import android.os.Bundle;
    import android.util.Log;
    import android.view.View;
    import android.widget.Button;
    import android.widget.TextView;
    import android.widget.Toast;

    import java.io.IOException;
    import java.io.InputStream;
    import java.util.ArrayList;
    import java.util.UUID;
```

```
public class MonitoringScreen extends Activity {

    private static final String TAG = "BlueTest5-MainActivity";
    private int mMaxChars = 50000;//Default
    private UUID mDeviceUUID;
    private BluetoothSocket mBTSocket;
    private ReadInput mReadThread = null;


    private boolean mIsUserInitiatedDisconnect = false;

    //TextView Test
    //Controls
    private TextView tempText, humText, movText, testView, altText,
presText;
    //TextView Test

    //Movement Commands
    //step 1: create an int array that would contain the button ids

    private static final int[] idArray = {R.id.fwdBtn, R.id.LBtn, R.id.RBtn,
R.id.sendBtn};

    //Step 2: create a button object
    private Button[] button = new Button[idArray.length];
    //Movement Commands


    private boolean mIsBluetoothConnected = false;

    private BluetoothDevice mDevice;

    private ProgressDialog progressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_monitoring_screen);

        ActivityHelper.initialize(this);

        Intent intent = getIntent();
        Bundle b = intent.getExtras();
        mDevice = b.getParcelable(MainActivity.DEVICE_EXTRA);
        mDeviceUUID =
UUID.fromString(b.getString(MainActivity.DEVICE_UUID));
        mMaxChars = b.getInt(MainActivity.BUFFER_SIZE);
        Log.d(TAG, "Ready");

        //TextView Test
        tempText = (TextView) findViewById(R.id.tempData);
        humText = (TextView) findViewById(R.id.humData);
        movText = (TextView) findViewById(R.id.movData);
        testView = (TextView) findViewById(R.id.testView);
        altText = (TextView) findViewById(R.id.altData);
        presText = (TextView) findViewById(R.id.presData);
        //TextView Test

        //Movement Commands
        ArrayList<Integer> num = new ArrayList<Integer>();
```

```
            Button oneBtn, twoBtn, threeBtn, sendBtn;

            oneBtn = (Button)findViewById(R.id.fwdBtn);
            twoBtn = (Button)findViewById(R.id.LBtn);
            threeBtn = (Button)findViewById(R.id.RBtn);
            sendBtn = (Button)findViewById(R.id.sendBtn);

            //step 3: assigning button array object name "button" to the button
ids
            for (int i = 0; i < idArray.length; i++) {
                button[i] = (Button) findViewById(idArray[i]);

                //button[i].setText("it works");
                button[i].setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {

                        switch (v.getId()) {
                            case R.id.fwdBtn:
                                //Toast.makeText(getApplicationContext(),
"Forward", Toast.LENGTH_SHORT).show();
                                oneBtn.setTextColor(Color.RED);

                                num.add(11);

                                tempText.setText(" ");
                                humText.setText(" ");
                                movText.setText(" ");
                                testView.setText(" ");
                                altText.setText(" ");
                                presText.setText(" ");

                                oneBtn.setClickable(false);

                                break;
                            case R.id.LBtn:
                                //Toast.makeText(getApplicationContext(),
"Left", Toast.LENGTH_SHORT).show();

                                twoBtn.setTextColor(Color.RED);

                                num.add(12);

                                tempText.setText(" ");
                                humText.setText(" ");
                                movText.setText(" ");
                                testView.setText(" ");
                                altText.setText(" ");
                                presText.setText(" ");

                                twoBtn.setClickable(false);

                                break;
                            case R.id.RBtn:
                                //Toast.makeText(getApplicationContext(),
"Right", Toast.LENGTH_SHORT).show();
                                threeBtn.setTextColor(Color.RED);

                                num.add(13);

                                tempText.setText(" ");
```

```
                                    humText.setText(" ");
                                    movText.setText(" ");
                                    testView.setText(" ");
                                    altText.setText(" ");
                                    presText.setText(" ");

                                    threeBtn.setClickable(false);

                                    break;
                                case R.id.sendBtn:

                                    tempText.setText(" ");
                                    humText.setText(" ");
                                    movText.setText(" ");
                                    testView.setText(" ");
                                    altText.setText(" ");
                                    presText.setText(" ");

                                    for (int j = 0; j < num.size(); j++) {

                                        try {

mBTSocket.getOutputStream().write(num.get(j));

                                        } catch (IOException e) {
                                            // TODO Auto-generated catch block
                                            e.printStackTrace();
                                        }

                                        try{
                                            Thread.sleep(6000);
                                        }
                                        catch(Exception e)
                                        {
                                            e.printStackTrace();
                                        }
                                    }

                                    try {
                                        mBTSocket.getOutputStream().write(14);

                                    } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                    }


                                    num.clear();
                                    oneBtn.setClickable(true);
                                    twoBtn.setClickable(true);
                                    threeBtn.setClickable(true);

                                    oneBtn.setTextColor(Color.BLACK);
                                    twoBtn.setTextColor(Color.BLACK);
                                    threeBtn.setTextColor(Color.BLACK);

                                    break;
                            }
                        }
                    });
                }
```

```
            //Movement Commands
        }

        private class ReadInput implements Runnable {

            private boolean bStop = false;
            private Thread t;

            public ReadInput() {
                t = new Thread(this, "Input Thread");
                t.start();
            }

            public boolean isRunning() {
                return t.isAlive();
            }

            @Override
            public void run() {
                InputStream inputStream;

                try {
                    inputStream = mBTSocket.getInputStream();
                    while (!bStop) {
                        byte[] buffer = new byte[256];
                        if (inputStream.available() > 0) {
                            inputStream.read(buffer);
                            int i = 0;
                            /*
                             * This is needed because new String(buffer) is
taking the entire buffer i.e. 256 chars on Android 2.3.4
http://stackoverflow.com/a/8843462/1287554
                             */
                            for (i = 0; i < buffer.length && buffer[i] != 0;
i++) {
                            }
                            final String strInput = new String(buffer, 0, i);

                            /*
                             * If checked then receive text, better design would
probably be to stop thread if unchecked and free resources, but this is a quick
fix
                             */

                            tempText.post(new Runnable()
                            {
                                @Override
                                public void run()
                                {
                                    //TextView Test
                                    tempText.setText(" ");
                                    humText.setText(" ");
                                    movText.setText(" ");
                                    testView.setText(" ");
                                    altText.setText(" ");
                                    presText.setText(" ");

                                    tempText.append(strInput, 0, 6);
                                    humText.append(strInput, 7, 14);
                                    presText.append(strInput, 15,27);
                                    altText.append(strInput, 28,39);
```

```
                                movText.append(strInput, 40, 49);
                                //testView.append(strInput, 50, 80);


                                //TextView Test
                            }
                        });



                    }
                    Thread.sleep(500);
                }
            } catch (IOException e) {
// TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
// TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        public void stop() {
            bStop = true;
        }

    }

    private class DisConnectBT extends AsyncTask<Void, Void, Void> {

        @Override
        protected void onPreExecute() {
        }

        @Override
        protected Void doInBackground(Void... params) {

            if (mReadThread != null) {
                mReadThread.stop();
                while (mReadThread.isRunning())
                    ; // Wait until it stops
                mReadThread = null;

            }

            try {
                mBTSocket.close();
            } catch (IOException e) {
// TODO Auto-generated catch block
                e.printStackTrace();
            }

            return null;
        }

        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
            mIsBluetoothConnected = false;
            if (mIsUserInitiatedDisconnect) {
```

```
                    finish();
                }
            }

        }

        private void msg(String s) {
            Toast.makeText(getApplicationContext(), s,
Toast.LENGTH_SHORT).show();
        }

        @Override
        protected void onPause() {
            if (mBTSocket != null && mIsBluetoothConnected) {
                new DisConnectBT().execute();
            }
            Log.d(TAG, "Paused");
            super.onPause();
        }

        @Override
        protected void onResume() {
            if (mBTSocket == null || !mIsBluetoothConnected) {
                new ConnectBT().execute();
            }
            Log.d(TAG, "Resumed");
            super.onResume();
        }

        @Override
        protected void onStop() {
            Log.d(TAG, "Stopped");
            super.onStop();
        }

        @Override
        protected void onSaveInstanceState(Bundle outState) {
      // TODO Auto-generated method stub
            super.onSaveInstanceState(outState);
        }

        private class ConnectBT extends AsyncTask<Void, Void, Void> {
            private boolean mConnectSuccessful = true;

            @Override
            protected void onPreExecute() {
                progressDialog = ProgressDialog.show(MonitoringScreen.this,
"Hold on", "Connecting");// http://stackoverflow.com/a/11130220/1287554
            }

            @Override
            protected Void doInBackground(Void... devices) {

                try {
                    if (mBTSocket == null || !mIsBluetoothConnected) {
                        mBTSocket =
mDevice.createInsecureRfcommSocketToServiceRecord(mDeviceUUID);
                        BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                        mBTSocket.connect();
                    }
                } catch (IOException e) {
```

```
        // Unable to connect to device
                        e.printStackTrace();
                        mConnectSuccessful = false;
                    }
                    return null;
                }

                @Override
                protected void onPostExecute(Void result) {
                    super.onPostExecute(result);

                    if (!mConnectSuccessful) {
                        Toast.makeText(getApplicationContext(), "Could not connect
to device. Is it a Serial device? Also check if the UUID is correct in the
settings", Toast.LENGTH_LONG).show();
                        finish();
                    } else {
                        msg("Connected to device");
                        mIsBluetoothConnected = true;
                        mReadThread = new ReadInput(); // Kick off input reader
                    }

                    progressDialog.dismiss();
                }

            }


        }
```

Java: ActivityHelper:

```
package com.example.anysensormonitoring;

import android.app.Activity;

import android.content.SharedPreferences;

import android.content.pm.ActivityInfo;

import android.preference.PreferenceManager;


public class ActivityHelper {

   public static void initialize(Activity activity) {

        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(activity);


        String orientation = prefs.getString("prefOrientation", "Null");

        if ("Landscape".equals(orientation)) {


activity.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        } else if ("Portrait".equals(orientation)) {
```

```
activity.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        } else {

activity.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
        }
    }
}
```

PreferencesActivity:

```
package com.example.anysensormonitoring;


import java.util.Map;


import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;


public class PreferencesActivity extends PreferenceActivity implements
OnSharedPreferenceChangeListener {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityHelper.initialize(this);
// Using this for compatibility with Android 2.2 devices
    }


    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
String key) {
        Preference pref = findPreference(key);


        if (pref instanceof ListPreference) {
            ListPreference listPref = (ListPreference) pref;
            pref.setSummary(listPref.getEntry());
```

```
            ActivityHelper.initialize(this);
        }

        if (pref instanceof EditTextPreference) {
            EditTextPreference editPref = (EditTextPreference) pref;
            pref.setSummary(editPref.getText());
        }
    }

    @Override
    protected void onPause() {

PreferenceManager.getDefaultSharedPreferences(this).unregisterOnSharedPreferenceC
hangeListener(this);
        super.onPause();
    }

    @Override
    protected void onResume() {

PreferenceManager.getDefaultSharedPreferences(this).registerOnSharedPreferenceCha
ngeListener(this);
        Map<String, ?> keys =
PreferenceManager.getDefaultSharedPreferences(this).getAll();

        for (Map.Entry<String, ?> entry : keys.entrySet()) {
// Log.d("map values", entry.getKey() + ": " + entry.getValue().toString());
            Preference pref = findPreference(entry.getKey());
            if (pref != null) {
                pref.setSummary(entry.getValue().toString());
            }
        }

        super.onResume();
    }

}
```

Layout: activity_main:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="20dp">

        <Button
            android:id="@+id/search"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginLeft="90dp"
            android:text="Search" />

        <Button
            android:id="@+id/connect"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginLeft="50dp"
            android:text="Connect" />




    </LinearLayout>
    <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">
    </ListView>


</LinearLayout>
```

Layout: activity_monitoring_screen

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MonitoringScreen">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/testView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Test Box"
                android:textAlignment="center"
                android:textSize="24sp" />
        </LinearLayout>

        <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/movText"
            android:layout_width="37dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Movement"
            android:textAlignment="center"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/movData"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textAlignment="center"
            android:textSize="24sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/humTxt"
            android:layout_width="36dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Humidity"
            android:textAlignment="center"
            android:textSize="24sp" />

        <TextView
```

```
        android:id="@+id/humData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textAlignment="center"
        android:textSize="24sp" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/tempTxt"
        android:layout_width="36dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Temperature"
        android:textAlignment="center"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/tempData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textAlignment="center"
        android:textSize="24sp" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
```

```
            android:id="@+id/presTxt"
            android:layout_width="36dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Pressure"
            android:textAlignment="center"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/presData"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textAlignment="center"
            android:textSize="24sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/altTxt"
            android:layout_width="36dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Altitude"
            android:textAlignment="center"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/altData"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textAlignment="center"
```

```
            android:textSize="24sp" />
    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button
            android:id="@+id/fwdBtn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Forward" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/LBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Left" />

        <Button
            android:id="@+id/RBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_weight="1"
            android:text="Right" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/sendBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Send" />
    </LinearLayout>

  </LinearLayout>


</LinearLayout>
```

Variables

Android Phone (Android Studio)

Variables:

- **Robot Control Variables:**
  - forwardBtn – Forward Button
  - leftBtn – Left Button
  - rightBtn – Right Button
  - sendBtn – Send Button
- **Display Sensors Variables:**
  - altitude – Altitude Data Textview
  - humidity – Humidity Data Textview
  - pressure – Pressure Data Textview
  - temperature – Temperature Data Textview
- **Display Proximity Variable:**
  - position – Proximity Data Textview
- **Bluetooth Variables:**
  - bt – Bluetooth

Microcontroller

Variables:

- **Collect Sensor Data Variables:**
  - alt – Altitude Data float
  - humidity – Humidity Data float
  - pressure – Pressure Data float
  - temperature – Temperature Data float
  - proximity – Proximity Data float
- **Motor Control Variables:**
  - Pin 40 - Left Motor Signal
  - Pin 39 - Right Motor Signal

I/O Assignments

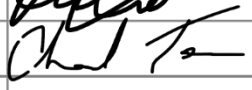
Name	I/O	Purpose	Routines
BMP280	Input	Collect Environmental Data	Subroutine 1
DHT11	Input	Collect Environmental Data	Subroutine 1
HC-SR04	Input	Proximity Sensor	Subroutine 1
HC-05	Input / Output	Bluetooth Module	

Document Approval

This section includes signatures of those that have written, reviewed or approved this High-Level Design deliverable for the Mini-Mars Rover

Author's Signature:

Your signature indicates that this document describes the Requirements of the Project Name Here and that this deliverable meets IEEE standards for documentation.

Name	Signature	Title/Department	Date
Alanzi, Nafa		Author	08/28/20
Tan, Chad Q.		Author	08/28/20
Walden, Dae'shaun		Author	08/28/20

Approvers' Signatures:

Your signature signifies that you agree with the high-level design presented in this document, and that it has been reviewed by appropriate personnel to ensure compliance with company and/or regulatory policies.

Name	Signature	Title/Department	Date
Andrew McNeely		Project Sponsor	08/28/20
		Process Owner	
Elizabeth Freije		Project Manager	08/28/20
		Application Architect	
		Quality Manager	

TEST METHODOLOGY/RESULTS

Overview

The project is designing and building a robot that can be commanded wirelessly through an android phone and collect data from the environment that is sent back to the phone. Most of the tests we are going to do are IEEE and NEMA standards.

Identification

This document provides a detailed test procedure that needs to be executed by the group members to evaluate the Mini-Mars Rover against the subset of application requirements that need to be electronically tested for this category

Testing Process

As previously mentioned, this document prescribes detailed test steps that need to be executed to test the requirements applicable for this category. The tests mentioned in this document are subject to change and modify anytime. The robot will be ready when passing all the tests shown in this document. The main test category is electricity, electronics, and programming debugging.

1 Test Procedure for Mini-Mars Rover

1.1 Requirements

The following table provides a reference to the requirements that need to be electronically tested within the Lab as outlined in the Approval Procedures document for the Product. The test cases that are used to check compliance to the requirements are cross-referenced in the table below.

Identifier #	Requirement Description	Source	Test Case #
Mars-Com.1	A software application created by the engineers to command the Mini-Mars Rover to function.		Mars-Coms.1

Table 1 - Applicable Requirements

1.2 Test Components

Table 2 provides the details of all the components required by the Lab to execute this test procedure. Based on the different test cases, different components may be required to execute different test cases.

#	Component	Component Details	Identifier
1	Android Phone	Includes the robot program to operate the Mini-Mars Rover installed.	Phone BT

Table 2 - Test Procedure: Components

1.3 Test Cases

This section discusses the various test cases that are needed to test the Mini-Mars Rover against the requirements mentioned above.

1.3.1 Test Case Mars-Coms.1

1.3.1.1 Purpose

The purpose of this test is to verify that the robot is communicating with the Android phone correctly, receiving all the commands from the phone, and operating in the correct order.

1.3.1.2 Test Setup

Equipment:	The following components are necessary for executing this test case:

[1] CBEFF header and signature requirements are considered to be out of scope for Template Generators.

	▪ Phone BT
Preparation:	▪ Connect the Android Phone's Bluetooth with the Microcontroller by writing code so it can function properly.

1.3.1.3 Test Process

Test Steps:	1. Press any of the available buttons in the order you wish the robot to operate within the program on the Android Phone. 2. Press "Send" once all the commands you wish to execute to the robot for operation. 3. Verify that all the data collected has been received from the robot, displayed on the Android phone program and that the robot had sent a signal back to the Android Phone that its working properly
Expected Result(s):	1. The test completes successfully by displaying the data collected from the environment and has displayed correctly on the Android Phone display.

Power point slides

Team Members & Advisor

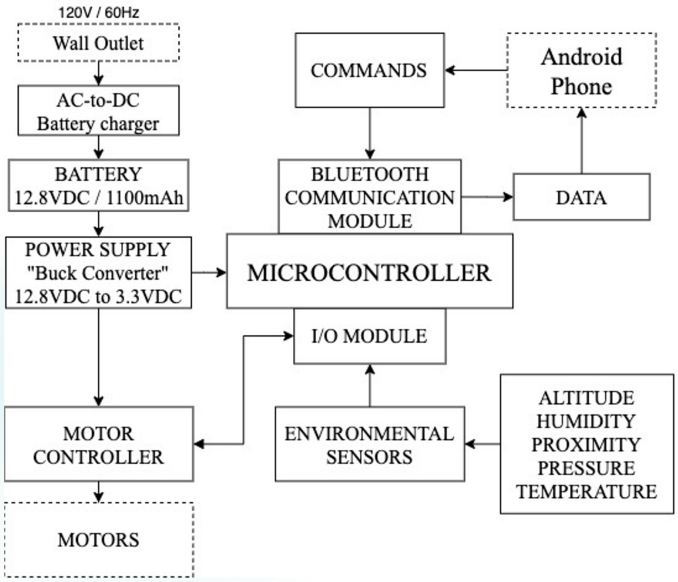
- **Nafa Alanzi** : Electrical Engineering Technology
- **Dae'Shaun Walden** : Electrical Engineering Technology
- **Chad Tan** : Computer Engineering Technology
- **Elizabeth Freije** : Faculty Advisor

Sponsor

- Andrew McNeely
- **Problem statement :**

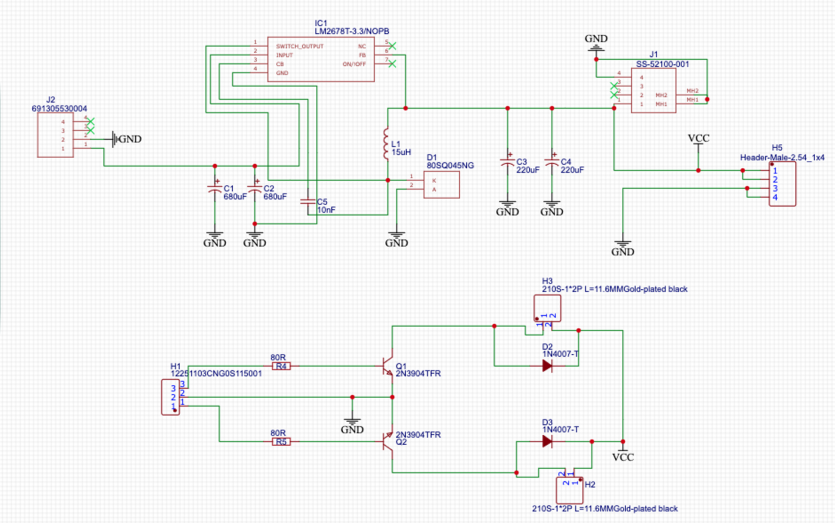
  "Construct a Robot to be used and demonstrated to the robotics club"

- **Project outcomes:**
- Robot Movement
- Collects 5 Environmental Data
- Operation time up to 1 Hour
- Charged through wall outlet
- Android phone controlled

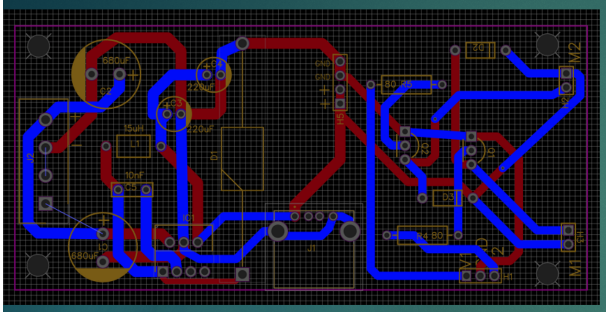


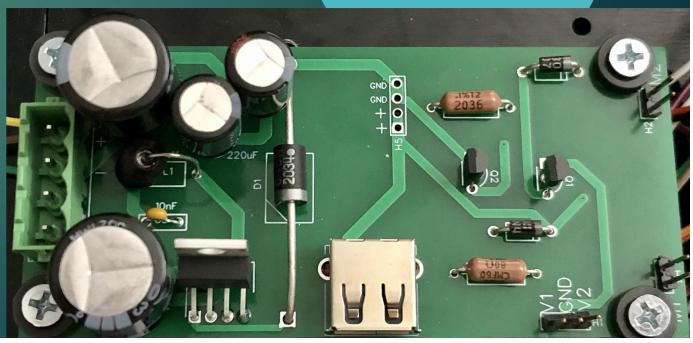
Hardware System Architecture

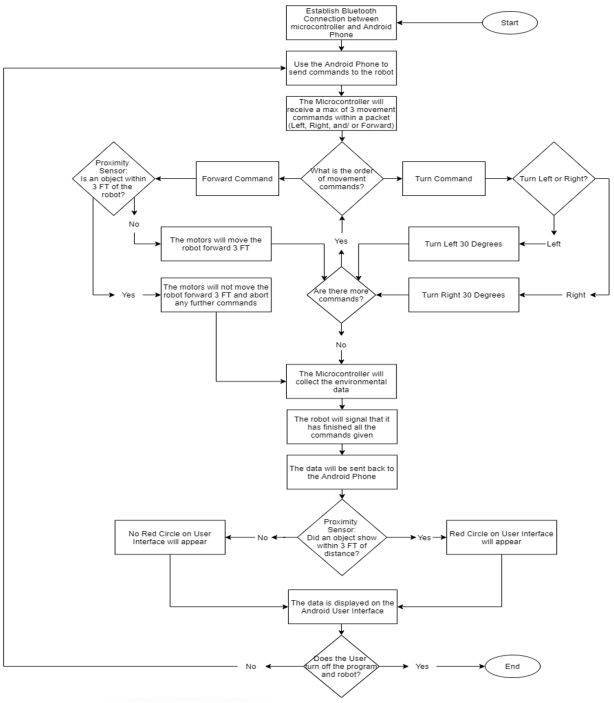
Power Circuit Schematic



Power Circuit PCB

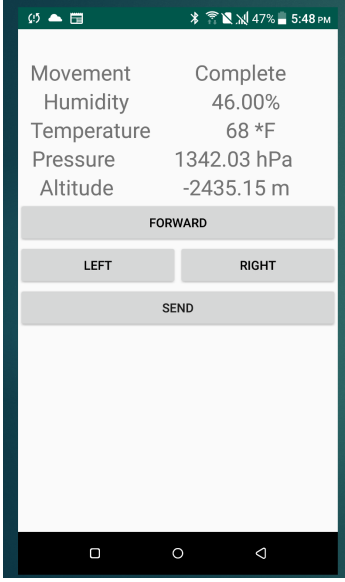






Code Flowchart

Android Phone Layout



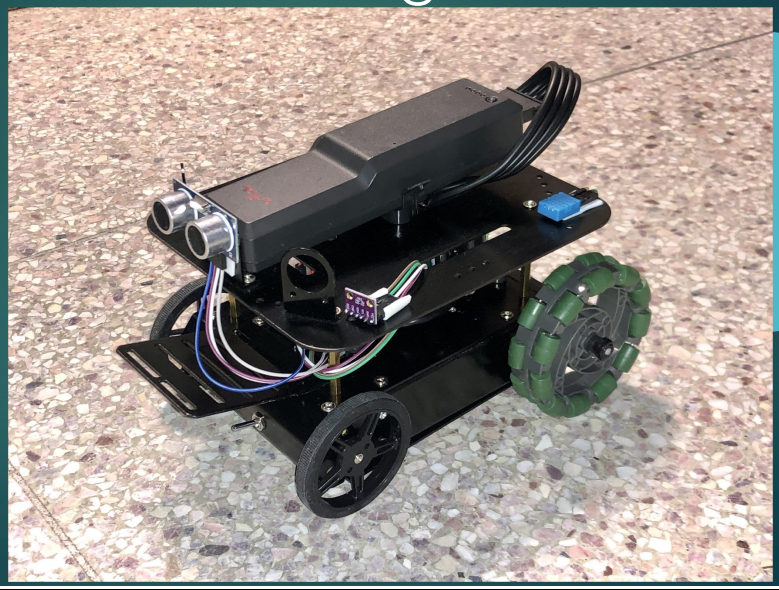
Inputs

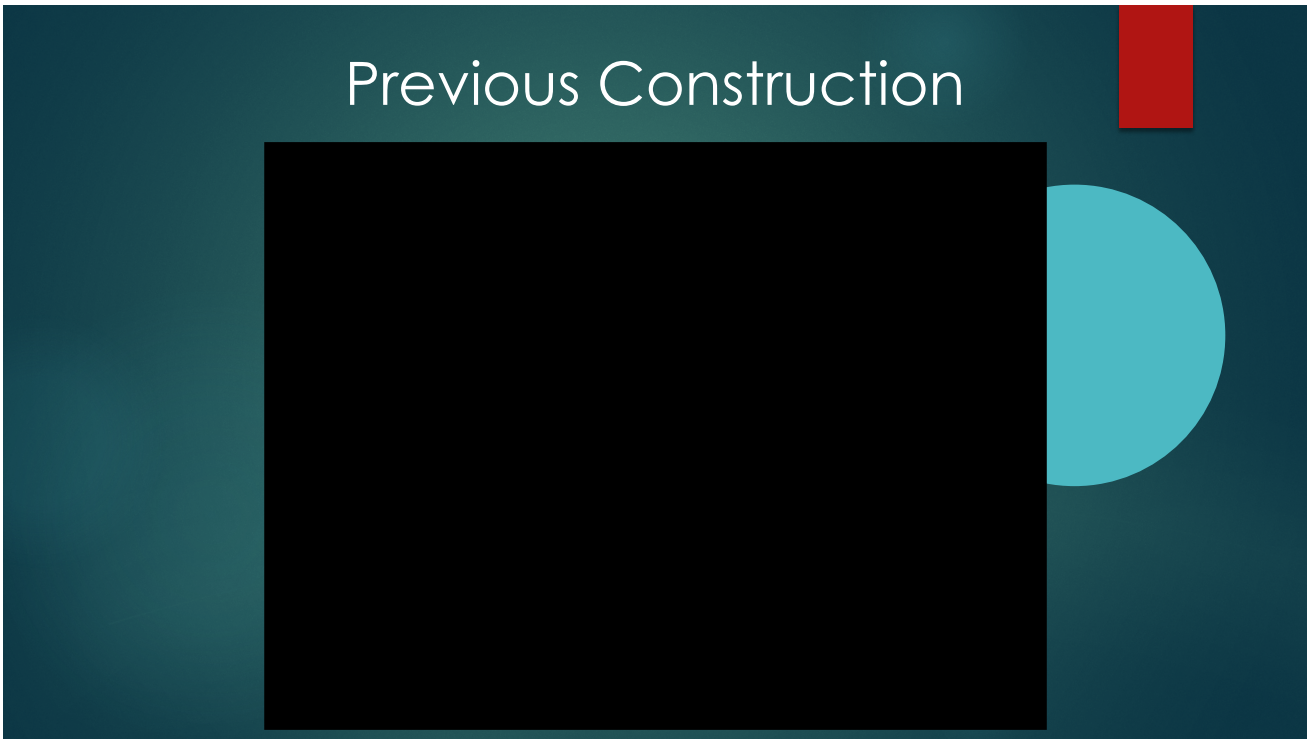
- Directional Keys
  - Each used once
  - Highlighted
  - Based on Order
- Send Button

Outputs

- Movement, Humidity, Temperature, Pressure, Altitude Labels

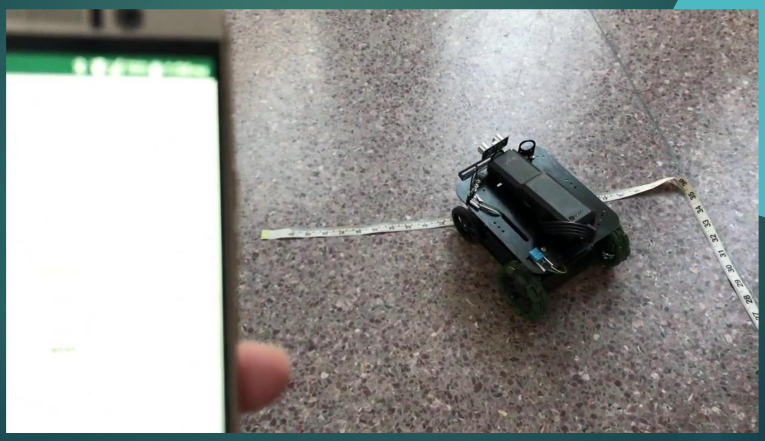
Robot Design

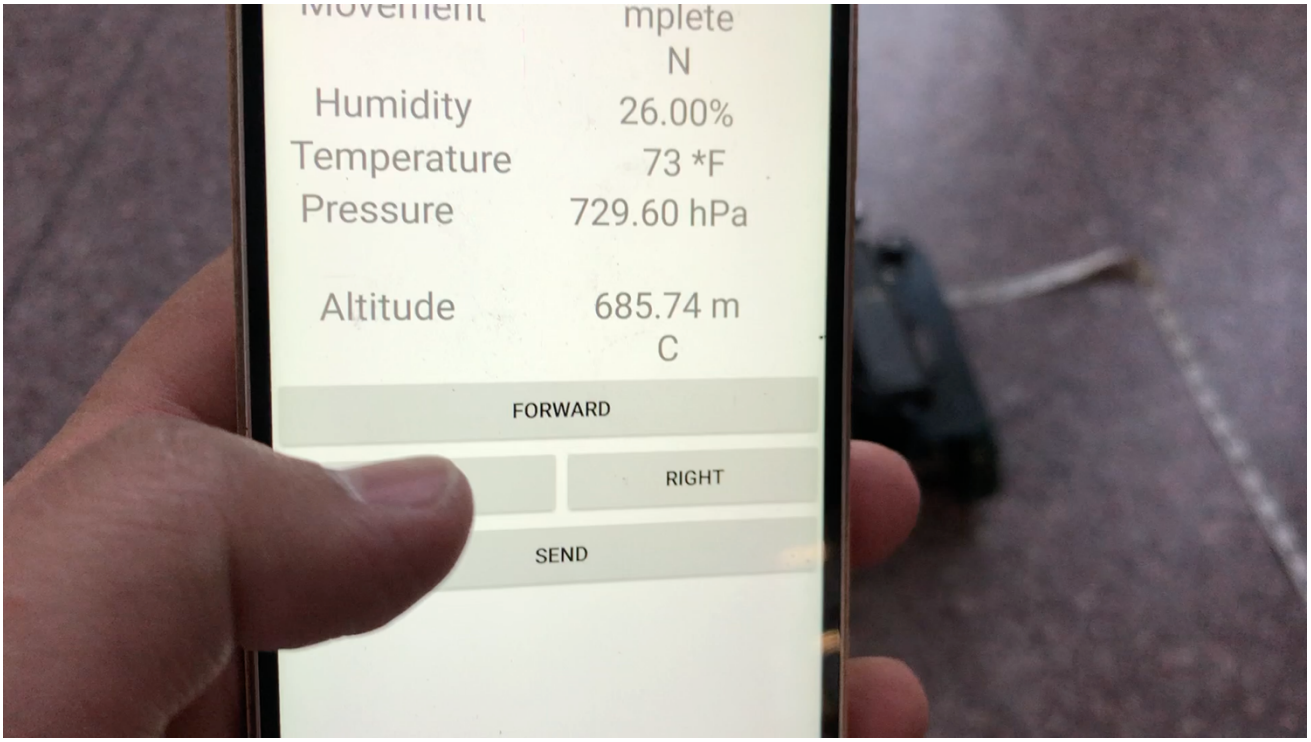



Previous Construction

Movement Demo

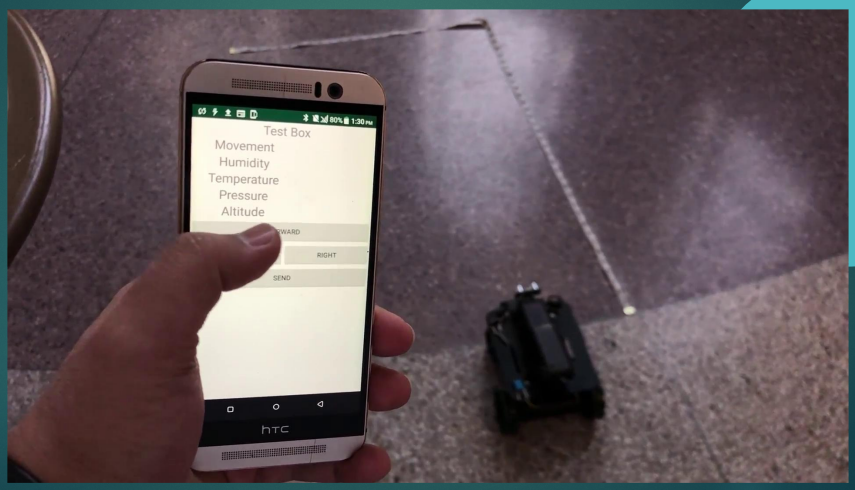
- 2 Movement Combo

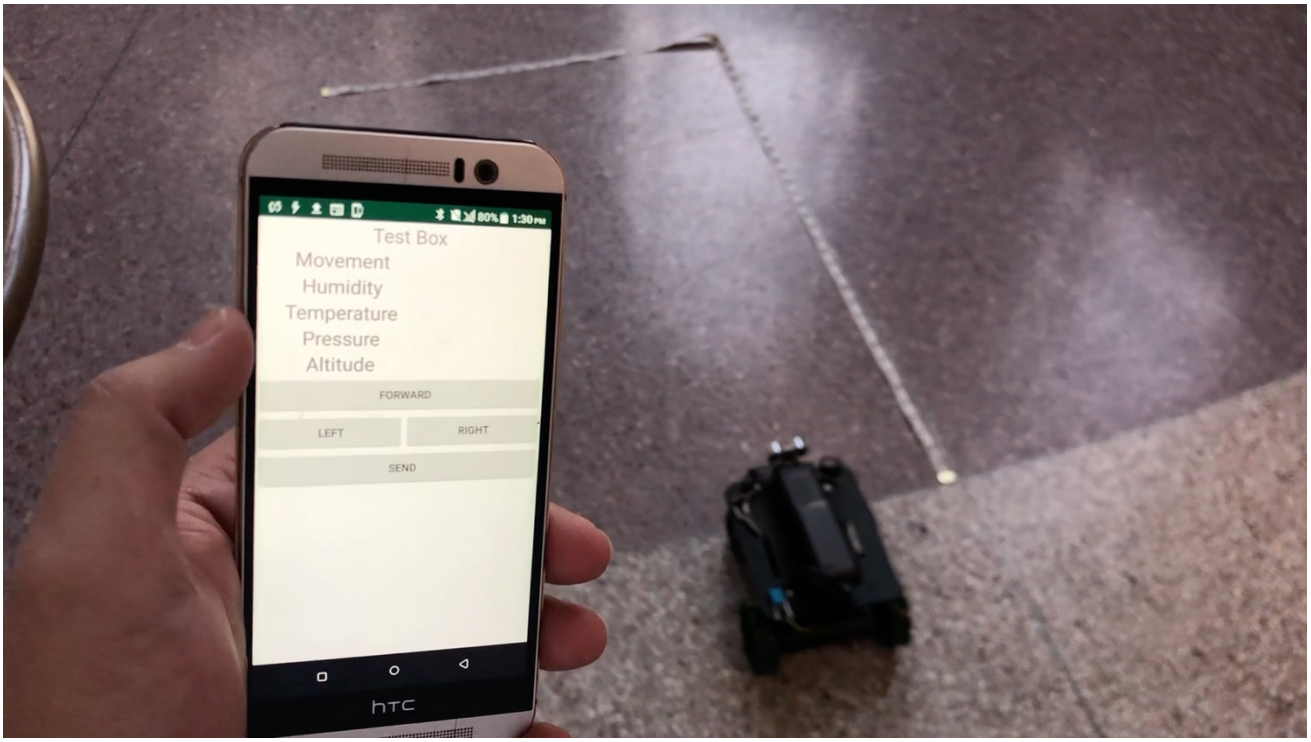




Movement Demo

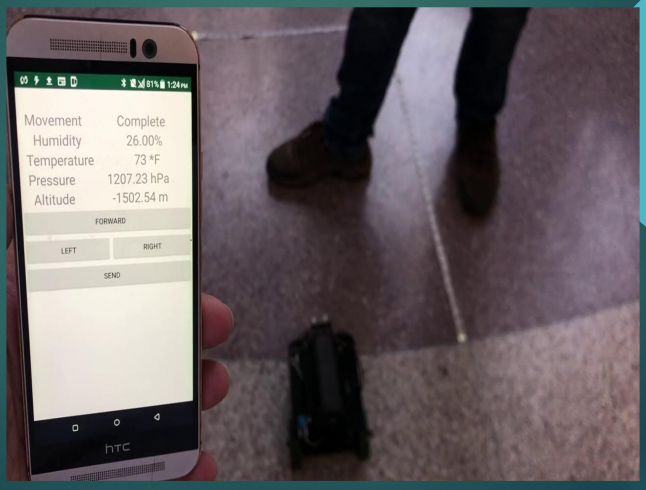
- 3 Movement Combo

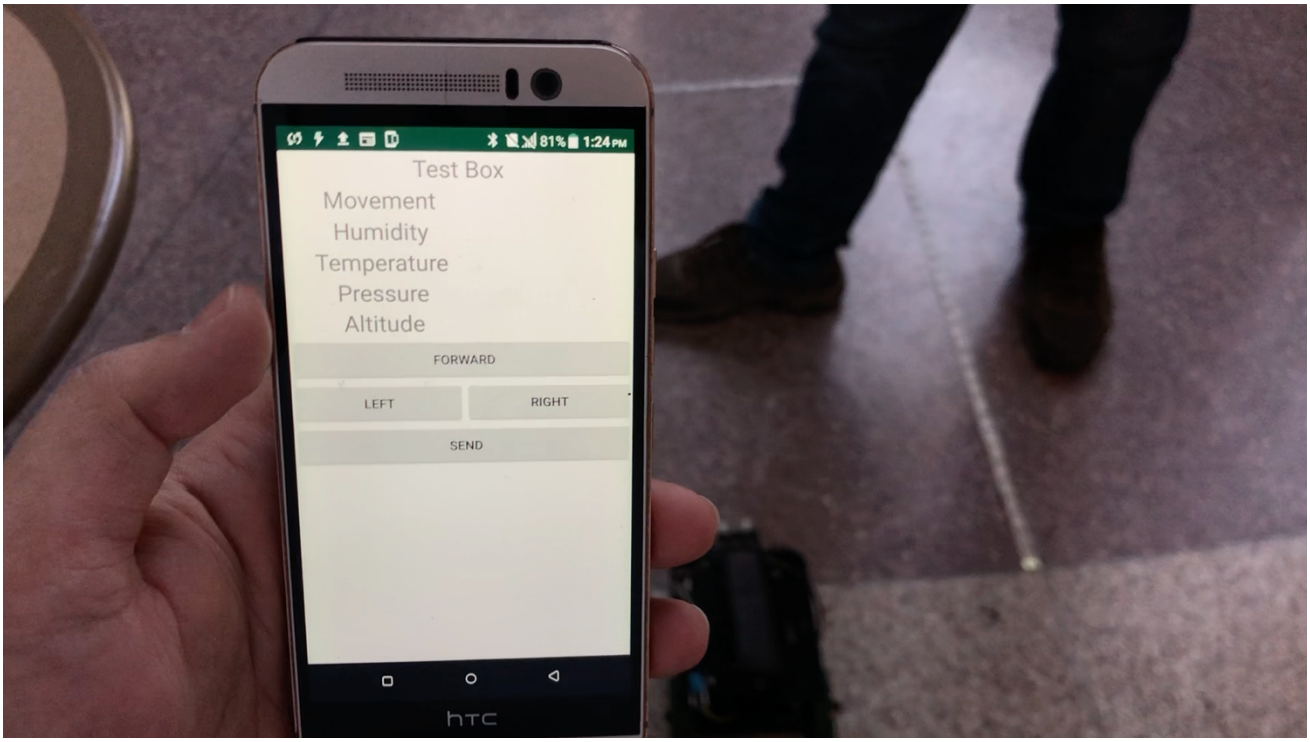


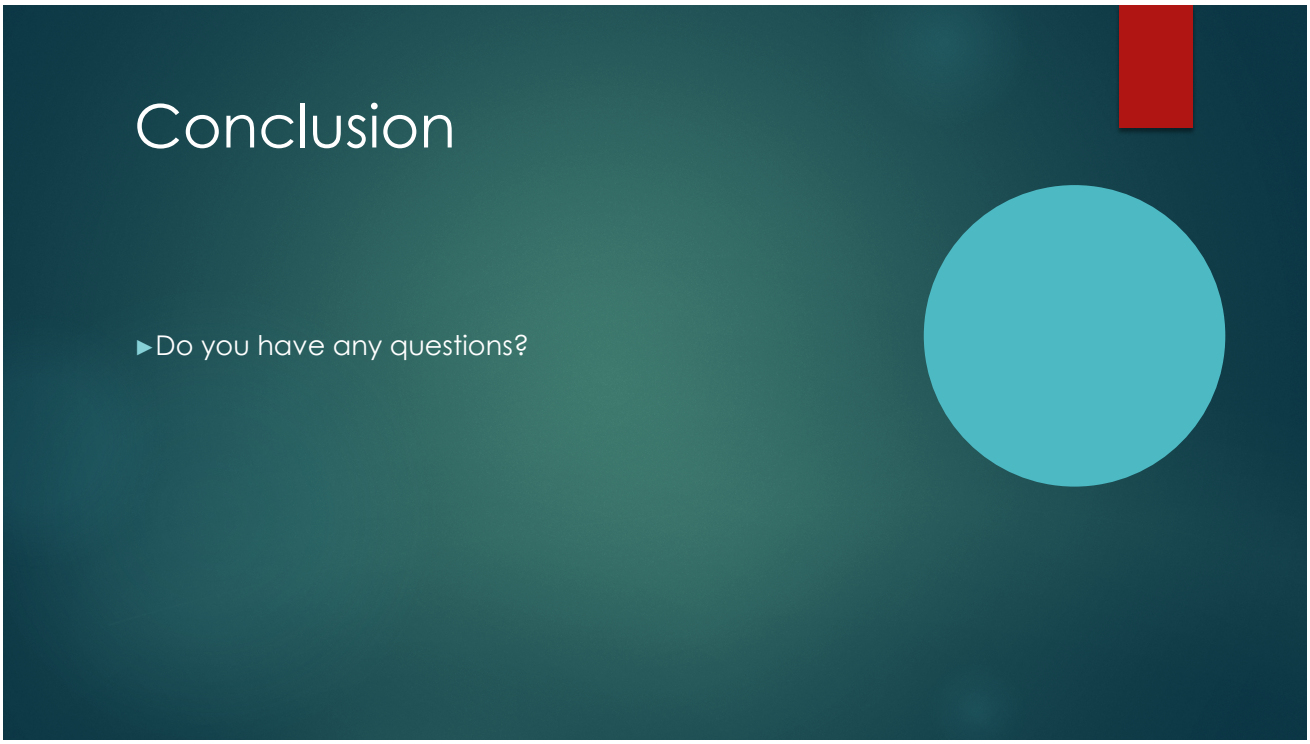


Movement Demo

- Forward Movement







CONCLUSION/INDIVIDUAL ASSESSMENT

We have designed and constructed a smaller scale version of the Mars Rover like the one from NASA. Importantly, considered the specifications of the customer when selecting and implementing the components to figure out what will work to fulfill this huge task. When we reach the end of the semester and the rover is completed, we can experience what it is like to operate a rover like the real-life NASA Mars rover.

NOTES

- Eliminated the solar system portion of the project.
- Switched from the VEX V5 robotics kit motors to commercial motors from our faculty advisor.
- Switched from the VEX V5 robotics kit frame to a commercial frame from our faculty advisor.
- Switched from regular batteries and to the battery from the VEX V5 robotics kit.
- Implemented the transistor circuit for the digital motor controller.

REFERENCES

1. **"HC-05 Bluetooth Module Interfacing with MSP-EXP430G2 TI Launchpad..," *ElectronicWings*. [Online]. Available: https://www.electronicwings.com/ti-launchpad/hc-05-bluetooth-module-interfacing-with-msp-exp430g2-ti-launchpad. [Accessed: 11-Oct-2020].**
2. **E. F. —, *How to Create Android App for Arduino Sensor Monitoring over Bluetooth*, 12-Sep-2020. [Online]. Available: https://www.electroniclinic.com/how-to-create-android-app-for-arduino-sensor-monitoring-over-bluetooth/. [Accessed: 15-Oct-2020].**
3. **E. F. —, "Android app development to control arduino over Bluetooth, Android studio," *Electronic Clinic*, 12-Nov-2020. [Online]. Available: https://www.electroniclinic.com/android-app-development-to-control-arduino-over-bluetooth-using-android-studio/. [Accessed: 20-Oct-2020].**
4. **"MSP-EXP432P401R LaunchPad and BMP280 barometric pressure sensor example," *Mikro blog*, 09-Mar-2018. [Online]. Available: http://www.mikroblog.net/msp/msp-exp432p401r-launchpad-bmp280-barometric-pressure-sensor-example.php. [Accessed: 05-Nov-2020].**
5. **A. A. Jabbaar, "Ultrasonic Sensor HC-SR04 with Arduino Tutorial," *Arduino Project Hub*. [Online]. Available: https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6. [Accessed: 20-Nov-2020].**
6. **"DHT11 Sensor Interfacing with MSP-EXP430G2 TI Launchpad: TI Laun..," *ElectronicWings*. [Online]. Available: https://www.electronicwings.com/ti-launchpad/dht11-sensor-interfacing-with-msp-exp430g2-ti-launchpad. [Accessed: 25-Sep-2020].**

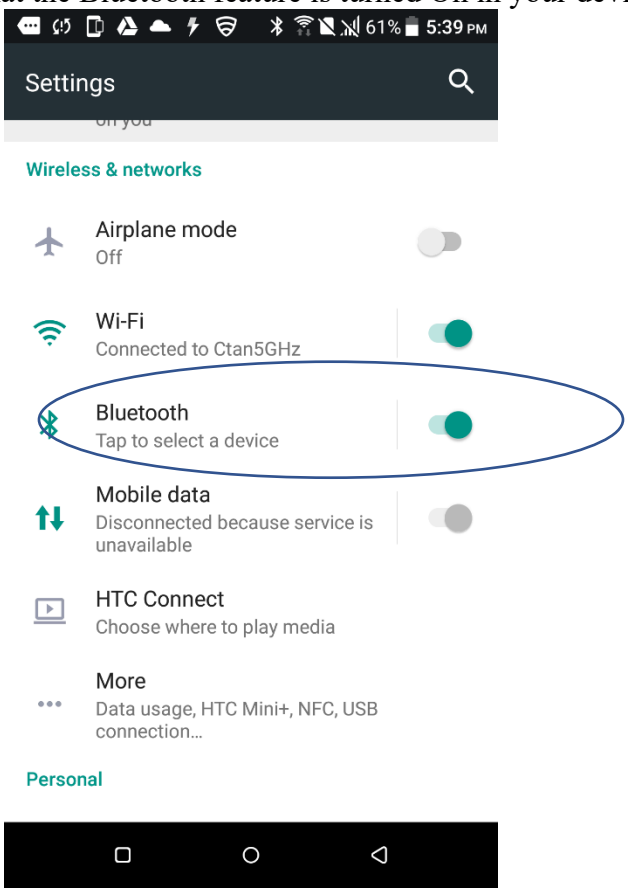
APPENDIXES

Final Project Specification (Signed)

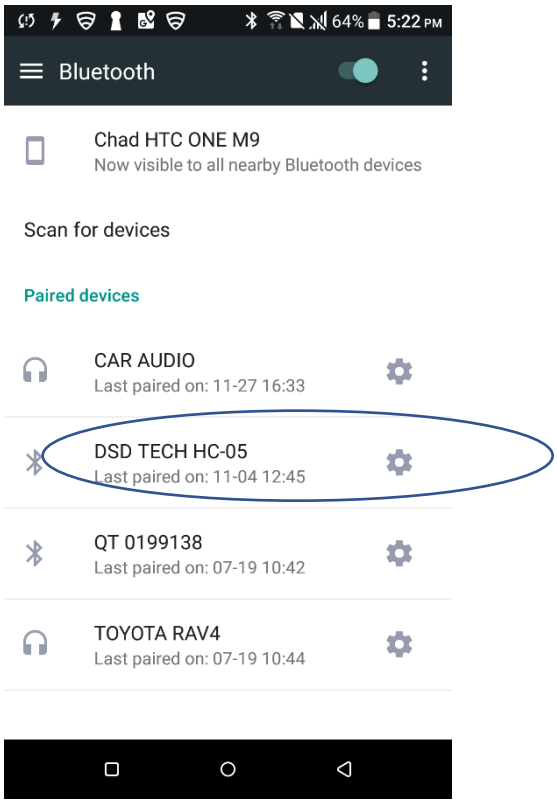
Project Manual

Connecting to the Mini-Mars Rover

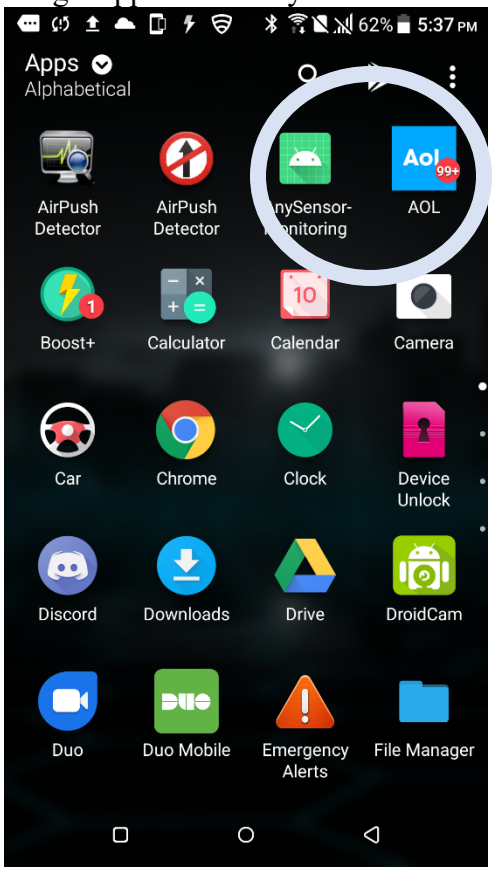
1. Open your Settings Application in you Android Device and access your Bluetooth Connection Tab (Be sure that the Bluetooth feature is turned On in your device):



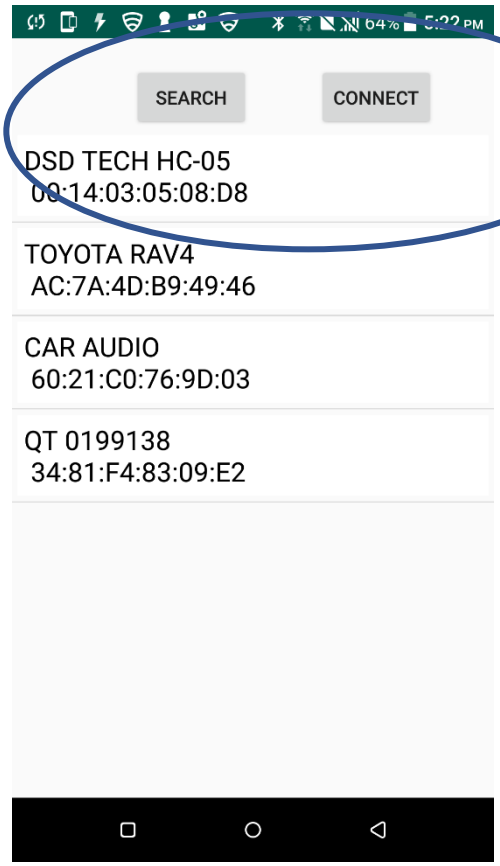
2. In the Bluetooth Tab Settings, you will scan for the “DSD TECH HC-05” and connect to it. The password for the device is “0000”:



3. Open the "AnySensorMonitoring" Application on your Android Device.



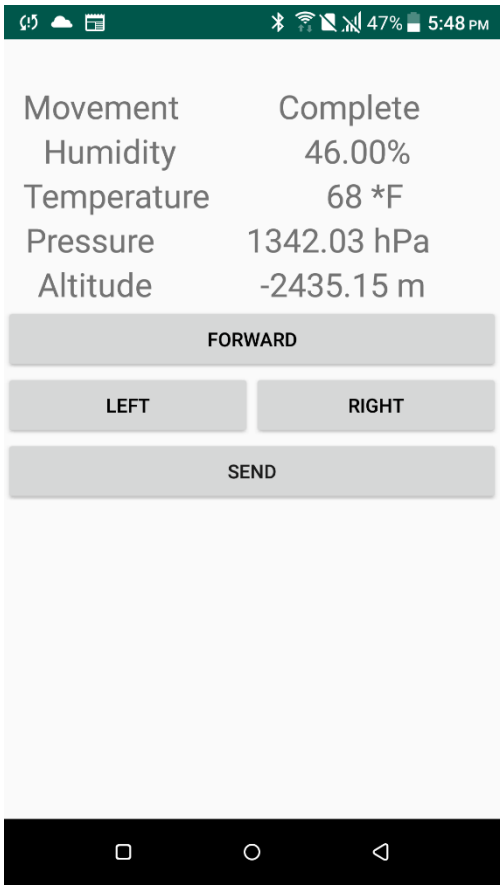
4. Within the application, press Search to open the list of Bluetooth devices connected. Select the "DSD TECH HC-05" amongst the list of Bluetooth devices and press "Connect":



Using the Application

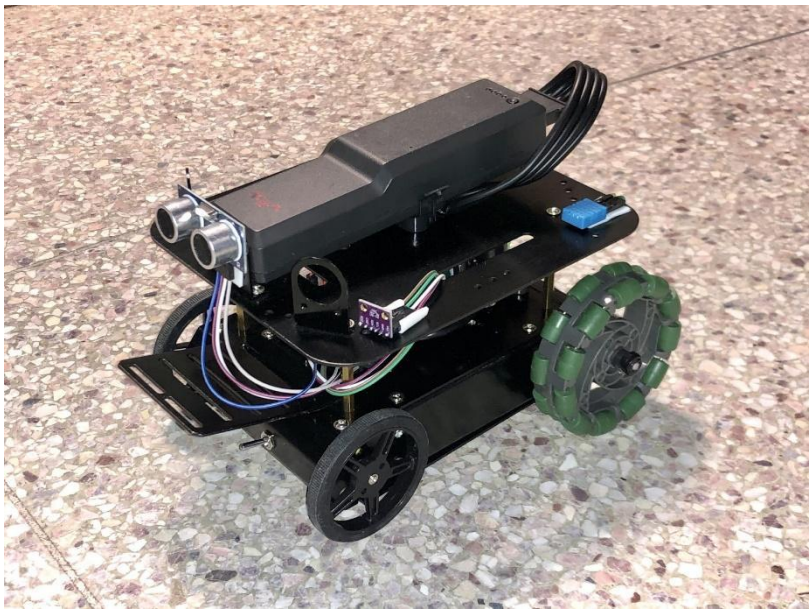
1. Select any movement command, to a max of three commands, in any order. One command cannot be pressed more than twice.
    - Forward = move Mini-Mars Rover Forward 3 FT
    - Left = turn Mini-Mars Rover Left 30°
    - Right = turn Mini-Mars Rover Right 30°

    Once selected, press “Send”.



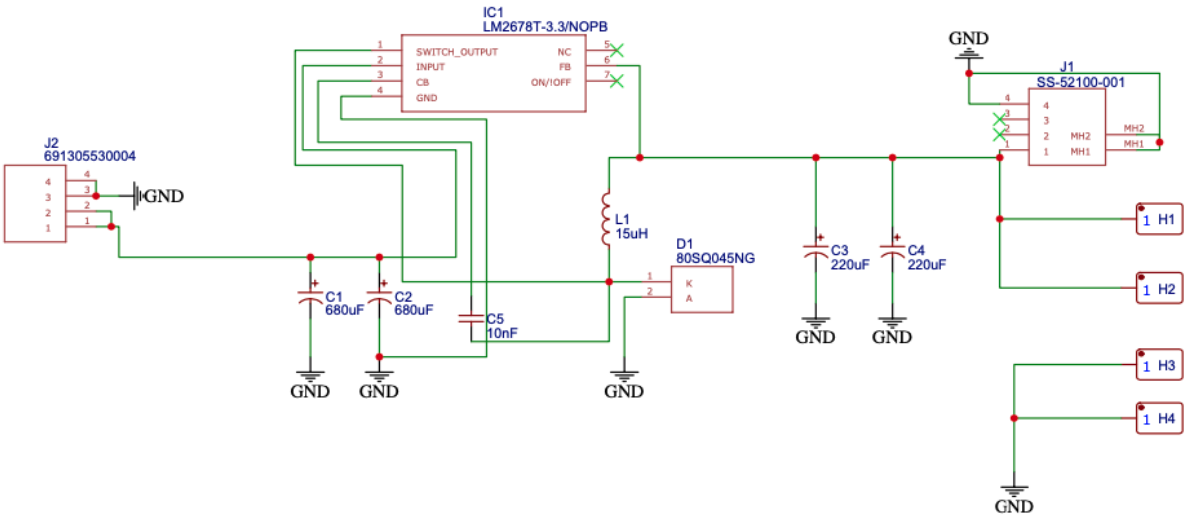
2. The Mini-Mars Rover will return the environmental data within each specific Data Labels. The “Error Message Box” will display errors.
    a. Error Messages:
        i. Movement Obstruction within 3 FT
        ii. Sensor Data Error

Final Project Design

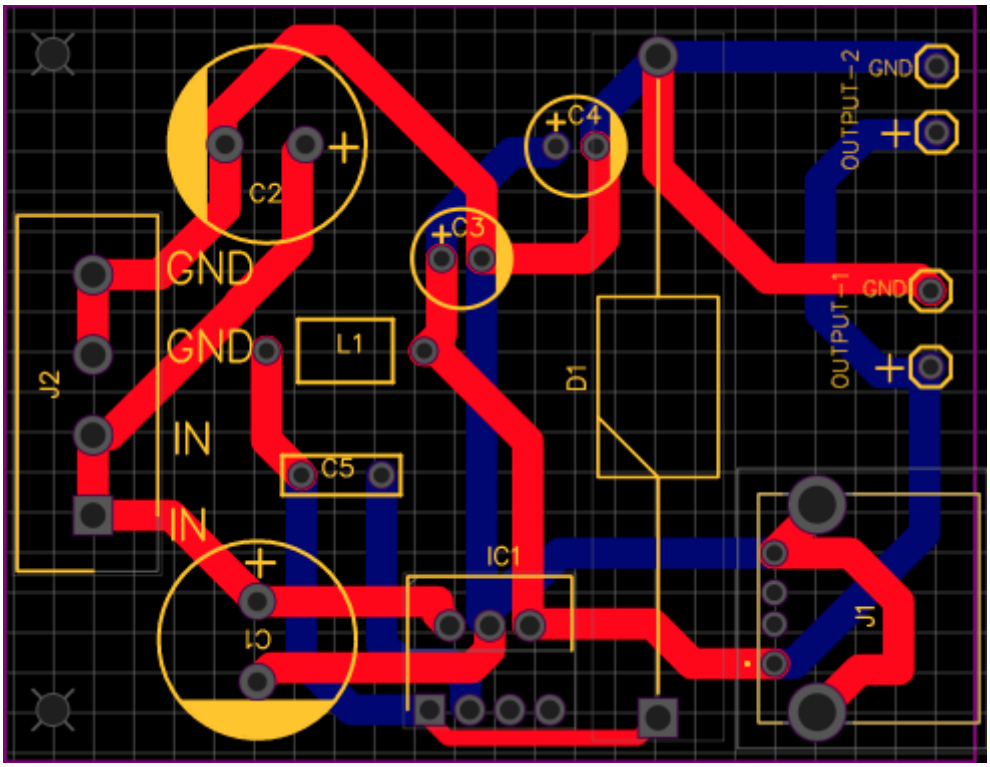


Schematics/Flowcharts

Power Supply

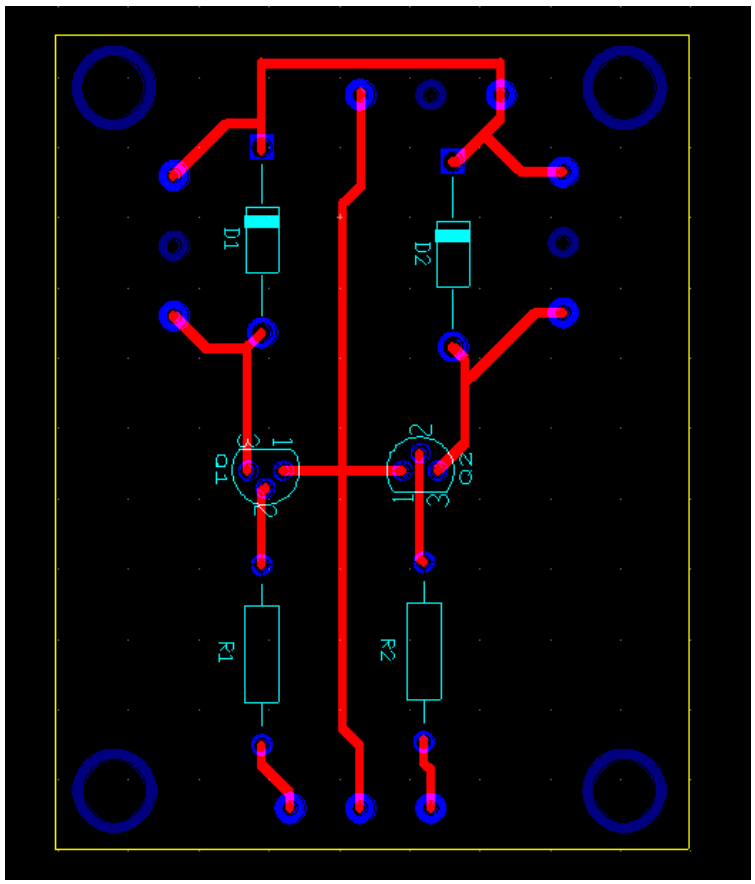


Power Supply PCB

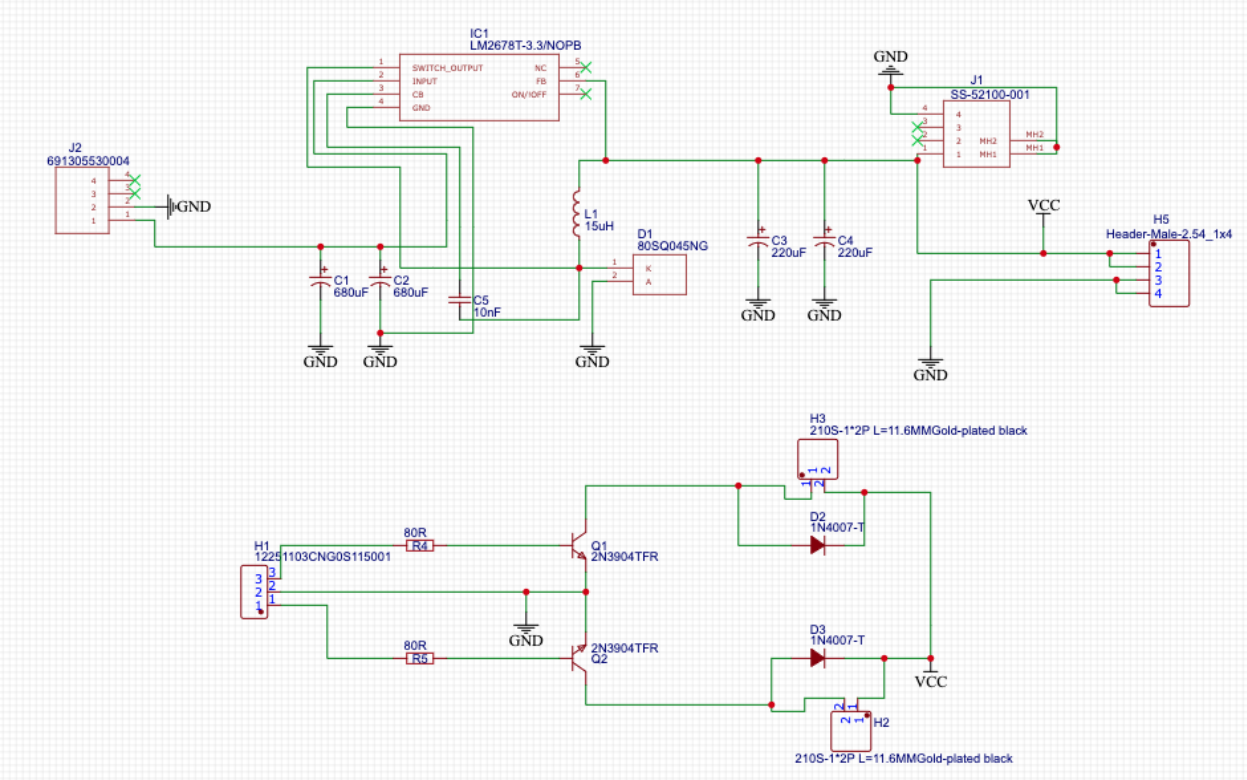


Transistor Circuit (80 Ω)

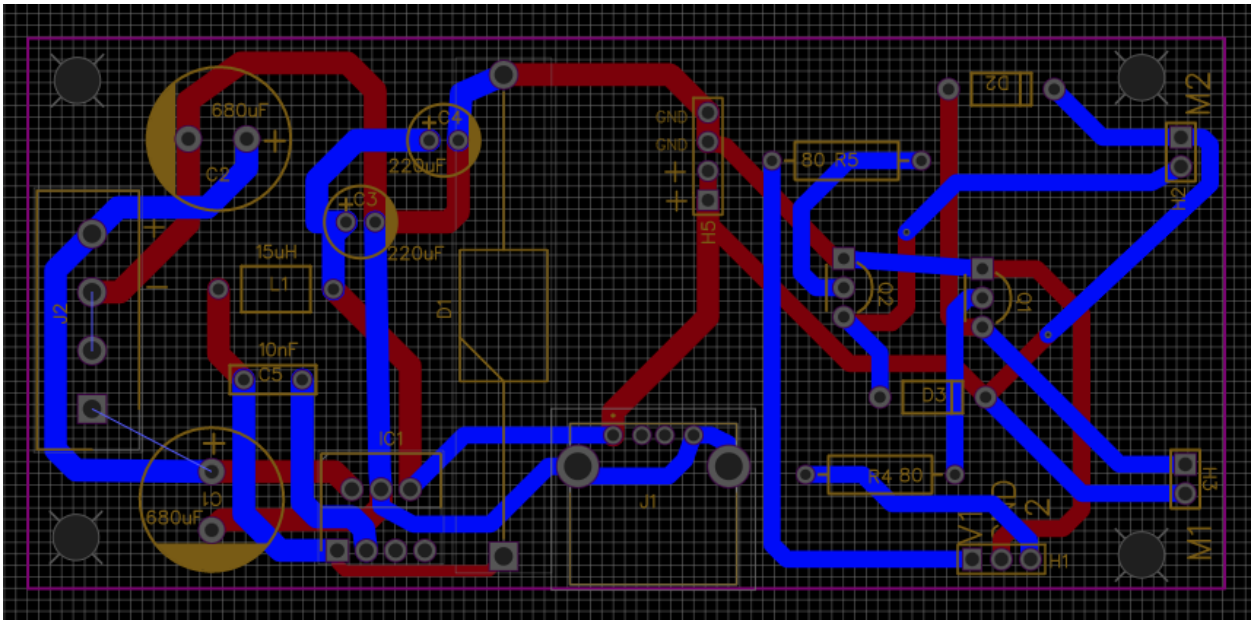
Transistor Circuit PCB



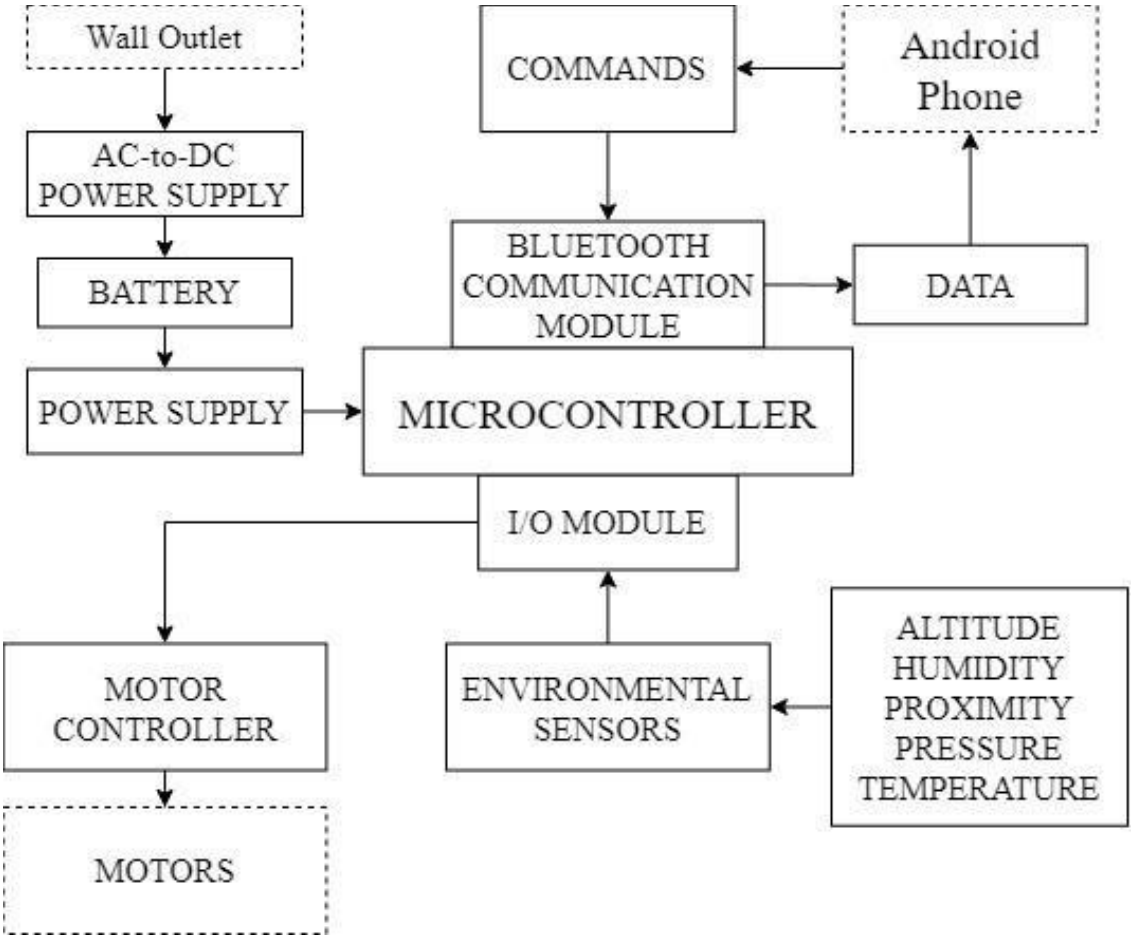
Power Circuit (Power Supply and Transistor circuit conjunction)

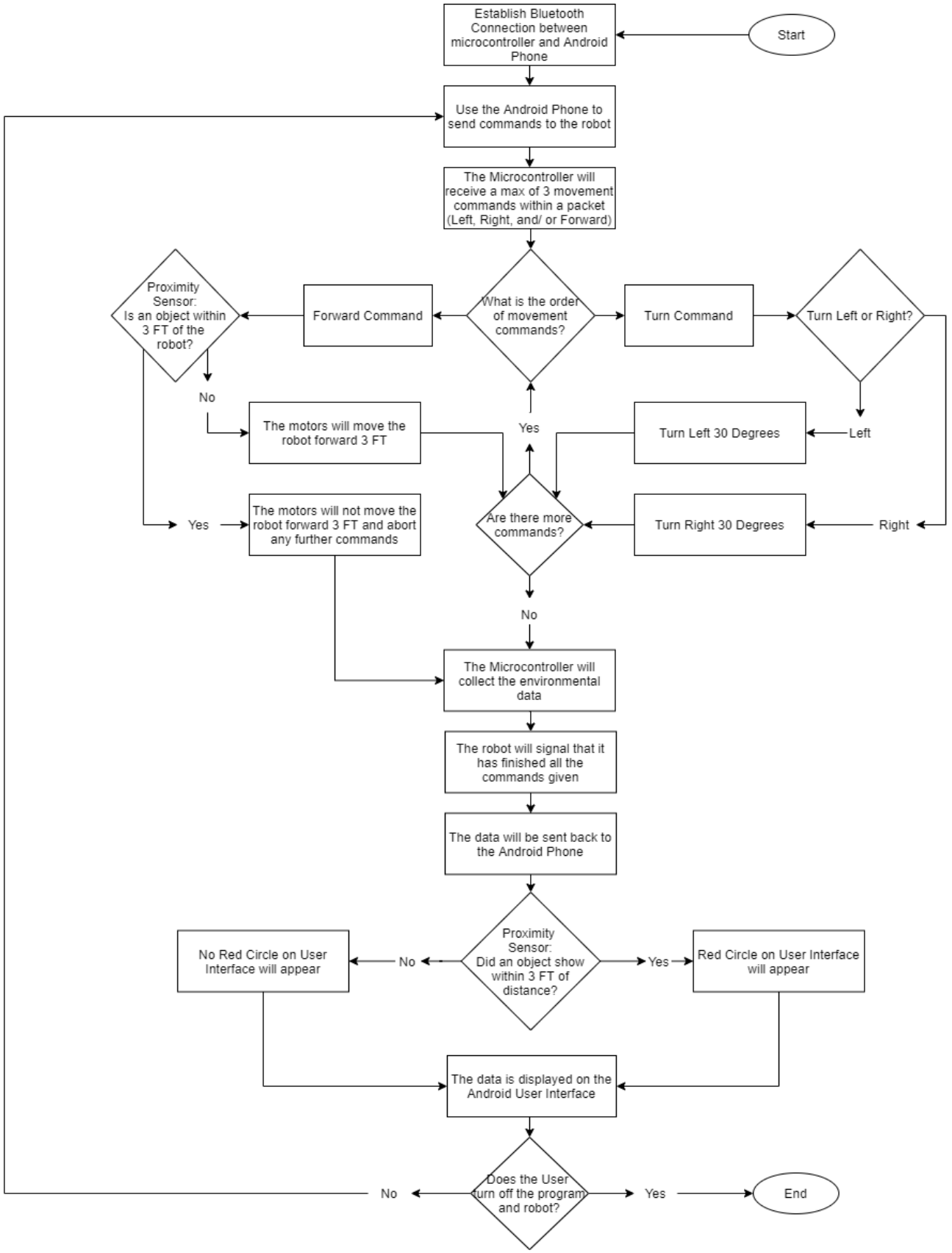


Power Circuit (Power Supply and Transistor circuit conjunction) PCB



System flow chart





Software code flowchart

Final Test Plan

Some major aspects we had to troubleshoot was the major voltage drop due to the transistor circuit which we resolved by changing the original transistor for a transistor that can handle more current which we then changed the resistor to match the calculations for the new motors we had received when previously the system was designed for the VEX V5 motors.

We improved on the project of the Mini Mars Rover by eliminating the solar panel system and implemented a transistor circuit that will serve a digital switch for the microcontroller to turn the motors on or off when the signal is received. We also eliminated the singular batteries for the battery from the VEX V5 robotics kit. The major change we implemented was to eliminate the motors and frame from the VEX V5 robotics kit because of them components being too proprietary for VEX. So, we received different types of commercial motors and a frame from one of our faculty advisors.

Overview

The project is designing and building a robot that can be commanded wirelessly through an android phone and collect data from the environment that is sent back to the phone. Most of the tests we are going to do are IEEE and NEMA standards.

Identification

This document provides a detailed test procedure that needs to be executed by the group members to evaluate the Mini-Mars Rover against the subset of application requirements that need to be electronically tested for this category

Testing Process

As previously mentioned, this document prescribes detailed test steps that need to be executed to test the requirements applicable for this category. The tests mentioned in this document are subject to change and modify anytime. The robot will be ready when passing all the tests shown in this document. The main test category is electricity, electronics, and programming debugging.

2 Test Procedure for Mini-Mars Rover

2.1 Requirements

The following table provides a reference to the requirements that need to be electronically tested within the Lab as outlined in the Approval Procedures document for the Product. The test cases that are used to check compliance to the requirements are cross-referenced in the table below.

Identifier #	Requirement Description	Source	Test Case #
Mars-Com.1	A software template created by the engineers to command the Mini-Mars Rover to function.		Mars-Coms.1

Table 1 - Applicable Requirements

2.2 Test Components

Table 2 provides the details of all the components required by the Lab to execute this test procedure. Based on the different test cases, different components may be required to execute different test cases.

#	Component	Component Details	Identifier
1	Android Phone	Includes the robot program to operate the Mini-Mars Rover installed.	Phone BT

Table 2 - Test Procedure: Components

2.3 Test Cases

This section discusses the various test cases that are needed to test the Mini-Mars Rover against the requirements mentioned above.

2.3.1 Test Case Mars-Coms.1

2.3.1.1 Purpose

The purpose of this test is to verify that the robot is communicating with the Android phone correctly, receiving all the commands from the phone, and operating in the correct order.

2.3.1.2 Test Setup

Equipment:	The following components are necessary for executing this test case:

[1] CBEFF header and signature requirements are considered to be out of scope for Template Generators

	▪ Phone BT
Preparation:	▪ Connect the Android Phone's Bluetooth with the Microcontroller by writing code so it can function properly.

2.3.1.3 *Test Process*

Test Steps:	4. Press any of the available buttons in the order you wish the robot to operate within the program on the Android Phone. 5. Press "Send" once all the commands you wish to execute to the robot for operation. 6. Verify that all the data collected has been received from the robot, displayed on the Android phone program and that the robot had sent a signal back to the Android Phone that its working properly
Expected Result(s):	2. The test completes successfully by displaying the data collected from the environment and has displayed correctly on the Android Phone display.

TROUBLESHOOTING/DESIGN IMPROVEMENTS

- Improve battery
    - Battery Automatically shuts off after 1 min no use
        - VEX batteries are proprietary
- Improve the code
    - Android Phone
        - Crashes when trying to display the data in the TextBoxes
            - Not due to interference from chassis
            - Possible deprecated code used
        - Was not able to implement an error message box
            - Example: Robot is obstructed from moving 3 FT Forward

Final Project Plan

Gantt Chart

Task Name	Duratio	Start	Finish	% Complete	Predecessors	Resource Names
Senior Design Project Phase 2	**85 days?**	**Mon 8/24/20**	**Fri 12/18/20**	**100%**		
Design	**67 days?**	**Sun 8/23/20**	**Sat 11/21/20**	**100%**		**Everyone**
Test Specifcation (signed)	6 days	Sun 8/23/20	Fri 8/28/20	100%		
Final Requirement Specification (signe	6 days	Sun 8/23/20	Fri 8/28/20	100%		
Revised LLD	7 days	Sun 8/23/20	Sun 8/30/20	100%		
Power Supply	**51 days?**	**Sat 8/29/20**	**Fri 11/6/20**	**100%**		**Nafa & Dae'Shaun**
Buck Converter	35 days	Mon 9/7/20	Fri 10/23/20	100%		**Nafa**
PCB Layout	30 days	Mon 9/14/20	Fri 10/23/20	100%		**Nafa**
Motors controller	14 days	Tue 10/6/20	Fri 10/23/20	100%		**Dae'Shaun**
Motors controller PCB Layout	14 days	Tue 10/6/20	Fri 10/23/20	100%		**Dae'Shaun**
Code	**56 days**	**Mon 9/7/20**	**Sat 11/21/20**	**100%**		**Chad Tan**
Sensor	**45 days**	**Mon 9/14/20**	**Fri 11/13/20**	**100%**		
BMP280	25 days	Fri 10/9/20	Fri 11/13/20	100%		
DHT11	15 days	Mon 9/14/20	Fri 10/2/20	100%		
JSN-SR04T	45 days	Mon 9/14/20	Fri 11/13/20	100%		
Android Phone Layout	45 days	Mon 9/14/20	Fri 11/13/20	100%		
Bluetooth Connection	**51 days**	**Mon 9/14/20**	**Sat 11/21/20**	**100%**		
Microcontroller	36 days	Mon 9/14/20	Sat 10/31/20	100%		
Android Phone	36 days	Mon 9/14/20	Sat 10/31/20	100%		
Motor Control	11 days	Mon 11/9/20	Sat 11/21/20	100%	36	
Verify	**26 days**	**Mon 11/9/20**	**Sun 12/13/20**	**100%**		**Everyone**
Code	**26 days**	**Mon 11/9/20**	**Sun 12/13/20**	**100%**		**Chad Tan**
Phone Control Layout	20 days	Mon 11/9/20	Fri 12/4/20	100%		
Testing power supply	26 days	Mon 11/9/20	Sun 12/13/20	100%		**Nafa & Dae'Shaun**
Motors	**14 days**	**Mon 11/2/20**	**Thu 11/19/20**	**100%**	**36,52**	**Everyone**
Power Supply Connection	13 days	Tue 11/3/20	Thu 11/19/20	100%	36	
Code Control Testing	14 days	Mon 11/2/20	Thu 11/19/20	100%		
Power Supply test	1 day	Mon 11/9/20	Mon 11/9/20	100%		**Nafa & Dae'Shaun**
Construct Robot	**27 days**	**Sun 11/1/20**	**Sat 12/5/20**	**100%**	**36,52,55,39**	**Everyone**
Building	17 days	Sun 11/1/20	Sat 11/21/20	100%		
Testing	12 days	Sun 11/22/20	Sat 12/5/20	100%		
Documentation & Presentation	9 days	Mon 12/7/20	Thu 12/17/20	100%		**Everyone**

Bill of Materials (BOM)

12/9/20

Mini Mars Rover BOM

Part Name	Part Number	Quantity	Description	Manufacturer	Cost	Total Cost
BMP280	2651	1	It is a sensor for temperature, Altitude and Pressure	Adafruit	$9.95	$9.95
DHT11	386	1	Humidity and Temperature Sensor	Adafruit	$5.00	$5.00
HC-05		1	Bluetooth Module	DSD Tech	$8.99	$8.99
Ranging Detector	HC-SR04	1	SainSmart HC-SR04 Ranging Detector Mod Distance Sensor (Blue)	SainStore Inc.	$4.95	$4.95
MSP432P401R		1	32-Bit Embedded Evaluation Board	Texas Instruments	$23.96	$23.96
LM2678T-3.3	LM2678T-3.3/NOPB	1	Switching Voltage Regulators	Texas Instruments	$ 5.84	$5.84
Terminal Blocks	691305530004	1	Pluggable Terminal Blocks	Wurth Elektronik	$ 1.69	$1.69
USB Connectors	SS-52100-001	1	USB Connectors USB 2.0 R Angle Recpt Type A solder	Bel	$ 0.46	$0.46
Schottky Diodes	80SQ045NG	1	Schottky Diodes & Rectifiers 8A 45V	ON Semiconductor	$ 0.78	$0.78
Capacitor 0.01uF	C317C103J1G5TA	1	Multilayer Ceramic Capacitors - Leaded 100V 0.01uF C0G 5%	KEMET	$ 0.52	$0.52
Inductors 15uH	5800-150-RC	1	Fixed Inductors 15uH 10%	Bourns	$ 1.10	$1.10
Capacitor 680uF	EEU-FS1J681S	2	Aluminum Electrolytic Capacitors - Radial Leaded 63VDC 680uF	Panasonic	$ 1.78	$3.56
Capacitor 220uF	EEU-TP1V221	2	Aluminum Electrolytic Capacitors - Radial Leaded 220uF 35volts	Panasonic	$ 1.27	$2.54
Resistor 80ohms	CMF6080R000BHEB	2	Metal Film Resistors - Through Hole 1W 80ohms .1%	Vishay / Dale	$ 0.78	$1.56
Diodes	1N4007-T	2	Rectifiers Vr/1000V Io/1A T/R	Diodes Incorporated	$ 0.19	$0.38
Headers	68604-804HLF	3	Headers & Wire Housings 68604-804HLF-BS SR VT TH HDR 1*4	Amphenol FCI	$ 0.40	$1.20
NPN Transistor	2N3904TF	2	Bipolar Transistors - BJT NPN 40V 200mA	ON Semiconductor / Fairchild	$ 0.27	$0.54
PCB		1	Buck converter + digital switchboard	jlcpcb	$ 22.00	$22.00

Total Cost	$95.02

Provided parts

Part Name	Part Number	Quantity	Description	Manufacturer	Cost	Total Cost
Pirate	ROB0022	1	4WD Mobile Robot Kit for Arduino with Bluetooth 4.0	DFRobot	$79.00	$79.00
V5 Robot Battery	276-4811	1	V5 Robot Battery Li-Ion 1100mAh	Innovation First International, Inc.	$54.99	$54.99
Omni Directional Wheel	276-3526	1	3.25" Omni-Directional Wheel (4-Pack)	Innovation First International, Inc.	$43.99	$43.99
Karcy Gear Motor	FYQ0001	1	Karcy Gear Motor TT Geared Motor for Smart Car Robot Yellow Dual Shaft for Car Chassis Pack of 2	Karcy	$6.29	$6.29

Total Cost	$184.27

Project Poster

IUPUI | SCHOOL OF ENGINEERING AND TECHNOLOGY
A PURDUE UNIVERSITY SCHOOL
Indianapolis

Mini Mars Rover

Nafa Alanzi (1), Dae'Shaun Walden (1), Chad Tan (2)

(1) Electrical Engineering Technology | (2) Computer Engineering Technology

Abstract: *The project is to design and construct a robot that can be commanded wirelessly through an Android phone and collect data from the environment that is sent back to the Android phone for observation. The brain and power of the rover is a combination of a microcontroller, a power supply buck converter, and a transistor motor control circuit which act as a digital switch. The rover is also a front wheel drive rover operating off two motors to enable the rover to move.*

Future

The future of this project after us is for the customer to show the rover to the robotics club, future IUPUI students, and to create a future senior design improvement project.

Background

We are creating a Mini Mars Rover that has the same characteristics as the real NASA Mars Rover but on a lower scale. The purpose of this project is to put IUPUI engineering students' skills on display.

Key Characteristics

- *Android controlled via Bluetooth*
- *Motor controlled*
- *Power supply/Buck converter implementation*
- *Digital transistor switch*
- *Five environmental data*
  - *Altitude*
  - *Temperature*
  - *Proximity (position)*
  - *Humidity*
  - *Pressure*

Visuals

Figure 1: Rover physical form

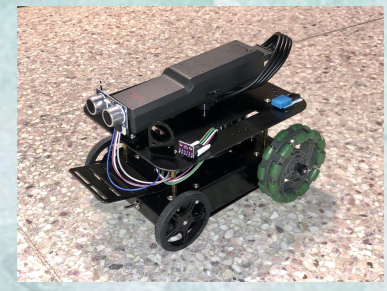
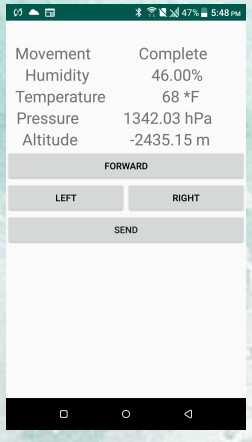


Figure 2: Android phone app



Presentation slides – In Appendixes folder

Weekly Progress Reports – In Appendixes folder

Name: Nafa Alanzi
Today's date: 09/4/2020
Week Number: 2
Total Hours for Week: 5hr

Expected Tasks to be Accomplished This Past Week

- **Review the progress in the project.**
- **Sign LLD document.**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Review the progress in the project.**
- **Sing LLD document.**

Problems Encountered This Past Week and Impact on Project

- **Know who is going to sign our document.**

Expectations for Next Week

- **Sizing and finding a battery for the robot .**
- **Find a charger for the battery.**

Name: Nafa Alanzi
Today's date: 09/11/2020
Week Number: 3
Total Hours for Week: 12hr

Expected Tasks to be Accomplished This Past Week

- **Sizing and finding a battery for the robot .**
- **Find a charger for the battery.**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Sizing and finding a battery for the robot .**
- **Find a charger for the battery.**

Problems Encountered This Past Week and Impact on Project

- **Finding a low budget battery and charger.**

Expectations for Next Week

- **Design Buck converter PCB layout.**

Name: Nafa Alanzi
Today's date: 09/18/2020
Week Number: 4
Total Hours for Week: 10hr

Expected Tasks to be Accomplished This Past Week

- **Design Buck converter PCB layout.**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Design Buck converter PCB layout.**

Problems Encountered This Past Week and Impact on Project

- **Changing the buck converter input from 12V to 12.8V .**

Expectations for Next Week

- **Design a 12.8V Buck converter.**
- **Design PCB layout.**

Name: Nafa Alanzi
Today's date: 09/25/2020
Week Number: 5
Total Hours for Week: 15hr

Expected Tasks to be Accomplished This Past Week

- Design a 12.8V Buck converter.
- Design PCB layout.

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Design a 12.8V Buck converter.

Problems Encountered This Past Week and Impact on Project

- PCB layout ports " terminal block and USB .

Expectations for Next Week

- Buck converter design adjustment
- PCB layout.

Name: Nafa Alanzi
Today's date: 10/06/2020
Week Number: 6
Total Hours for Week: 12hr

Expected Tasks to be Accomplished This Past Week

- Buck converter design adjustment
- PCB layout check.

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Buck converter design adjustment

Problems Encountered This Past Week and Impact on Project

- PCB layout tracing .

Expectations for Next Week

- Gannt chart.
- Order components

Name: Nafa Alanzi
Today's date: 10/09/2020
Week Number: 7
Total Hours for Week: 15hr

Expected Tasks to be Accomplished This Past Week

- Gannt chart.
- Order components

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Gannt chart.

Problems Encountered This Past Week and Impact on Project

- Order components and find fast shipping.

Expectations for Next Week

- Order components
- Prepare for the midterm presentation

Name: Nafa Alanzi
Today's date: 10/16/2020
Week Number: 8
Total Hours for Week: 10hr

Expected Tasks to be Accomplished This Past Week

- Order components
- Prepare for the midterm presentation

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Prepare for the midterm presentation .

Problems Encountered This Past Week and Impact on Project

- Order components and find fast shipping.

Expectations for Next Week

- Order components
- Print PCB layout.

Name: Nafa Alanzi
Today's date: 10/23/2020
Week Number: 9
Total Hours for Week: 8hr

Expected Tasks to be Accomplished This Past Week

- Order components
- Print PCB layout

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Order components
- Print PCB layout.

Problems Encountered This Past Week and Impact on Project

- Building the VEX robot.

Expectations for Next Week

- Building the VEX robot
- Soldering the components in the PCB

Name: Nafa Alanzi
Today's date: 10/30/2020
Week Number: 10
Total Hours for Week: 10hr

Expected Tasks to be Accomplished This Past Week

- Building the VEX robot.
- Soldering the components in the PCB.

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Building the VEX robot.
- Soldering the components in the PCB.

Problems Encountered This Past Week and Impact on Project

- Building the VEX.

Expectations for Next Week

- Continue building the VEX robot
- Testing the buck converter.

Name: Nafa Alanzi
Today's date: 11/6/2020
Week Number: 11
Total Hours for Week: 14hr

Expected Tasks to be Accomplished This Past Week

- Continue building the VEX robot
- Testing the buck converter

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Continue building the VEX robot
- Testing the buck converter

Problems Encountered This Past Week and Impact on Project

- Testing the buck converter . I need to change the input polarity.

Expectations for Next Week

- Change the input polarity.
- Continue building the VEX robot

Name: Nafa Alanzi
Today's date: 11/13/2020
Week Number: 12
Total Hours for Week: 10hr

Expected Tasks to be Accomplished This Past Week

- Continue building the VEX robot
- Change the input polarity in the buck converter.
- Change Zoom meeting to in person meeting.

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Continue building the VEX robot.
- Change Zoom meting to in person meeting.

Problems Encountered This Past Week and Impact on Project

- VEX motors do not work with the power supply I made. They only work with the VEX controller.

Expectations for Next Week

- Demonstrate our progress in person.
- Continue building the VEX robot

Name: Nafa Alanzi
Today's date: 11/20/2020

Week Number: 15
Total Hours for Week: 10hr

Expected Tasks to be Accomplished This Past Week

- Continue building the VEX robot.
- Demonstrate our progress in person.
- Change Zoom meeting to in person meeting.

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Change Zoom meeting to in person meeting.
- Continue building the VEX robot.
- Demonstrate our progress in person .

Problems Encountered This Past Week and Impact on Project

- Finding motors replacement that work with the PCB I built.

Expectations for Next Week

- Continue building the VEX robot
- Correct the buck convert input polarity.
- Correct the motors circuit.

Name: Nafa Alanzi
Today's date: 11/27/2020
Week Number: 14
Total Hours for Week: 20hr

Expected Tasks to be Accomplished This Past Week

- Continue building the VEX robot
- Correct the buck convert input polarity.
- Correct the motors circuit.
- Finding motors replacement that work with the PCB I built.

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Continue building the VEX robot
- Correct the buck convert input polarity.
- Finding motors replacement that work with the PCB I built.

Problems Encountered This Past Week and Impact on Project

- Correct the motors circuit due to the lack of resistor value availability.

Expectations for Next Week

- Combined the two-circuit buck converter and motor circuit.
- Correct the motors circuit.
- Order a new PCB.
- Test the circuit and the robot operation.

Name: _____Chad Tan_____
Today's date: _____09/04/20_____
Week Number: _____2_____
Total Hours for Week: _____4_____

Expected Tasks to be Accomplished This Past Week

- Research more into Bluetooth connection for the microcontroller and phone
- Type Code
- Test connection

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research more into Bluetooth connection for the microcontroller and phone (3 HR)
- Type partial Code (1 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix errors in code before I can test the connection

Expectations for Next Week

- More work into Bluetooth connectivity between the microcontroller and the phone

Name: _____Chad Tan_____
Today's date: _____09/011/20_____
Week Number: _____3_____
Total Hours for Week: _____3_____

Expected Tasks to be Accomplished This Past Week

- Research more into Bluetooth connection for the microcontroller and phone
- Type Code
- Test connection

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research more into Bluetooth connection for the microcontroller and phone (1 HR)
- Type Code (2 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix errors in code
- Not enough time to focus on project this week because of schedule and other small home projects

Expectations for Next Week

- More work into Bluetooth connectivity between the microcontroller and the phone
- Possibly take break from Bluetooth and focus on coding the sensors to the microcontroller

Name: _____Chad Tan_____
Today's date: _____09/19/20_____
Week Number: _____4_____
Total Hours for Week: _____6_____

Expected Tasks to be Accomplished This Past Week

- More work into Bluetooth connectivity between the microcontroller and the phone
- Possibly take break from Bluetooth and focus on coding the sensors to the microcontroller

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research more into Bluetooth connection for the microcontroller and phone (1 HR)
- Researched code for some sensors (2 HR)
- Type Code (3 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix errors in code

Expectations for Next Week

- More work into Bluetooth connectivity between the microcontroller and the phone
- Possibly take break from Bluetooth and focus on coding the sensors to the microcontroller

Name: _____Chad Tan_____
Today's date: _____09/26/20_____
Week Number: _____5_____
Total Hours for Week: _____3_____

Expected Tasks to be Accomplished This Past Week

- More work into Bluetooth connectivity between the microcontroller and the phone
- Possibly take break from Bluetooth and focus on coding the sensors to the microcontroller

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Researched code for some sensors (1.5 HR)
- Type Code (1.5 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
  - Need to fix a few errors in code written so far
- Transition to a new job in the past week, so had less time to focus on project because of new schedule

Expectations for Next Week

- Write some or Finish the code for the sensors

Name: _____Chad Tan_____
Today's date: _____10/02/20_____
Week Number: _____6_____
Total Hours for Week: _____4_____

Expected Tasks to be Accomplished This Past Week

- Write some or Finish the code for the sensors

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Researched code for some sensors (2 HR)
- Type Code (2 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
  - Need to fix a few errors in code written so far
- Finished most of the code that I was suppose to, but had to attend a wedding out of state for the whole weekend

Expectations for Next Week

- Finish the rest of the sensor coding
- Work on LLD
- Possibly jump on Bluetooth Connection

Name: _____Chad Tan_____
Today's date: _____10/09/20_____
Week Number: _____7_____
Total Hours for Week: _____6.5_____

Expected Tasks to be Accomplished This Past Week

- Finish the rest of the sensor coding
- Work on LLD
- Possibly jump on Bluetooth Connection
- Updating Gantt Chart

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Researched code for some sensors (1 HR)
- Researched code for Bluetooth (1.5)
- Type Code (2.5 HR)
- Update Gantt Chart (1 HR)
- Work on LLD (0.5 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
    - Data not being read correctly

Expectations for Next Week

- Work on Bluetooth Connection and Android Design

Name: _____Chad Tan_____
Today's date: _____10/016/20_____
Week Number: _____8_____
Total Hours for Week: _____5_____

Expected Tasks to be Accomplished This Past Week

- Work on Bluetooth Connection and Android Design

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Type Bluetooth & Android Layout Code (2 HR)
- Research Code (2 HR)
- Update Gantt Chart (0.5 HR)
- Work on LLD (0.5 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far

Expectations for Next Week

- Work on Bluetooth Connection and Android Design
- Work on presentation

Name: _____Chad Tan_____
Today's date: _____10/23/20_____
Week Number: _____9_____
Total Hours for Week: _____5_____

Expected Tasks to be Accomplished This Past Week

- Work on Bluetooth Connection and Android Design
- Work on presentation

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research Code (2 HR)
- Update Gantt Chart (0.5 HR)
- Worked on presentation (2 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
- Difficulty finding solutions to code

Expectations for Next Week

- Work on Bluetooth Connection and Android Design

Name: _____Chad Tan_____
Today's date: _____10/30/20_____
Week Number: _____10_____
Total Hours for Week: _____9_____

Expected Tasks to be Accomplished This Past Week

- Work on Bluetooth Connection
- Send and Receive Data on Android Phone
- Android Design

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research Code (4 HR)
- Bluetooth Connection (4 HR)
- Android Design (1 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
- Difficulty finding solutions to code
- Discovered that the wire connection was wrong

Expectations for Next Week

- Send and Receive Data on Android Phone
- Android Design

Name: _____Chad Tan_____
Today's date: _____11/6/20_____
Week Number: _____11_____
Total Hours for Week: _____6_____

Expected Tasks to be Accomplished This Past Week

- Send and Receive Data on Android Phone
- Android Design

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research Code (2.5 HR)
- Wrote code for sending and receiving data on Android Phone (2.5 HR)
- Android Design (1 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
- Difficulty finding solutions to code

Expectations for Next Week

- Update Android Layout to customer specs
- Clean Sensor Data
- Start on LED-Movement Test code

Name: _____Chad Tan_____
Today's date: _____11/13/20_____
Week Number: _____12_____
Total Hours for Week: _____5_____

Expected Tasks to be Accomplished This Past Week

- Update Android Layout to customer specs
- Clean Sensor Data
- Start on LED-Movement Test code

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research Code (1.5 HR)
- Lookup solutions and other methods to LED Test Plan (2 HR)
- Wrote Code (1.5 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
- Difficulty finding solutions to code

Expectations for Next Week

- Finish LED Test Plan and implement with motors

Name: _____Chad Tan_____
Today's date: _____11/20/20_____
Week Number: _____13_____
Total Hours for Week: _____6_____

Expected Tasks to be Accomplished This Past Week

- Finish LED Test Plan and implement with motors

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research Code (2 HR)
- Lookup solutions and other methods to LED Test Plan (2 HR)
- Wrote Code (2 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
- Difficulty finding solutions to code

Expectations for Next Week

- Implement the remaining sensors
- Display the rest of the sensor data on the android phone

Name: _____Chad Tan_____
Today's date: _____11/27/20_____
Week Number: _____14_____
Total Hours for Week: _____8_____

Expected Tasks to be Accomplished This Past Week

- Implement the remaining sensors
- Display the rest of the sensor data on the android phone

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- Research Code (3 HR)
- Lookup solutions and other methods to LED Test Plan (2 HR)
- Wrote Code (3 HR)

Problems Encountered This Past Week and Impact on Project

- Issues with code
    - Need to fix a few errors in code written so far
- Difficulty finding solutions to code
- Android Phone sometimes crashes when receiving back the data to be displayed on the screen

Expectations for Next Week

- Clean up layout
- Figure out crashing issue on Android Phone

Name: Dae'Shaun Walden

Today's date: 8-28-20

Week Number: 1

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Final Requirement Specification (signed)**
- **Gantt Chart**
- **Test Specification (signed)**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed Final Requirement Specification (signed)**
- **Completed Gantt Chart**
- **Completed Test Specification (signed)**

Problems Encountered This Past Week and Impact on Project

- **Setting a meeting time with respect to Nafa**

Expectations for Next Week

- **Complete week 2 progress report**

Name: Dae'Shaun Walden

Today's date: 9-3-20

Week Number: 2

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Week 2 progress report**
- **Confirm meeting times**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 2 progress report**

Problems Encountered This Past Week and Impact on Project

- **Getting back in rhythm post-covid emergency exit**

Expectations for Next Week

- **Schedule mid-semester design review**
- **Week 3 progress report**
- **Continue project**

Name: Dae'Shaun Walden

Today's date: 9-11-20

Week Number: 3

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Schedule mid-semester design review**
- **Week 3 progress report**
- **Meet Prof. Goodman about the motors**
- **Update the parts list**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 3 progress report**
- **Met Prof. Goodman about the motors**
- **Updated the parts list**

Problems Encountered This Past Week and Impact on Project

- **Finding websites to order from since the school does not prefer amazon**
- **Testing motors for the voltage and wattage**
- **Finding the proper lithium battery with an AC adapter**

Expectations for Next Week

- **Complete week 4 progress report**
- **Work on circuit board**
- **Continue project**

Name: Dae'Shaun Walden

Today's date: 9-19-20

Week Number: 4

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Week 4 progress report**
- **Schedule mid-semester design review**
- **Order parts from bill of materials**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 4 progress report**
- **Ordered parts from bill of materials**
- **Tested motors**

Problems Encountered This Past Week and Impact on Project

- **Confirming the bill of materials to be ordered**
- **Slight change on LLD**
- **Adjust PBC layout**

Expectations for Next Week

- **Complete week 5 progress report**
- **Wait for the bill of materials items to come in**
- **Review how to implement the VEX battery into the project**

Name: Dae'Shaun Walden

Today's date: 9-25-20

Week Number: 5

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 5 progress report**
- **Wait for the bill of materials items to come in**
- **Review how to implement the VEX battery into the project**
- **Set new meeting time**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 5 progress report**
- **Ordered bill of materials items**
- **Continuing the project in detail**

Problems Encountered This Past Week and Impact on Project

- **New meeting time with advisors**
- **Implementing VEX battery into project**

Expectations for Next Week

- **Complete week 6 progress report**
- **Get new meeting time set**
- **Continue project**

Name: Dae'Shaun Walden

Today's date: 10-2-20

Week Number: 6

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 6 progress report**
- **Pick up items from Craig**
- **Get new meeting time set**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Complete week 6 progress report**
- **Picked up items from Craig**
- **Got new meeting time which is now Thursdays**

Problems Encountered This Past Week and Impact on Project

- **Working with the circumstances of the campus due to covid-19**

Expectations for Next Week

- **Complete week 7 progress report**
- **Complete mid-semester Gantt chart**
- **Continue project**

Name: Dae'Shaun Walden

Today's date: 10-9-20

Week Number: 7

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 7 progress report**
- **Complete mid-semester Gantt chart**
- **Revise LLD**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Complete week 7 progress report**
- **Completed mid-semester Gantt chart**
- **Revised LLD**

Problems Encountered This Past Week and Impact on Project

- **Finishing buck converter**
- **Implement the transistor circuit in conjunction with the buck converter and motors**

Expectations for Next Week

- **Figure out the transistor circuit to implementation**
- **Complete week 8 progress report**
- **Set in person meeting times**
- **Start Presentation**

Name: Dae'Shaun Walden

Today's date: 10-17-20

Week Number: 8

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 8 progress report**
- **Set in person meeting times**
- **Start presentation**
- **Figure out the transistor circuit to implementation**
- **Update Gantt Chart**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Complete week 8 progress report**
- **Got in person times set**
- **Understand the totality of the transistor circuit**
- **Updated Gantt Chart**

Problems Encountered This Past Week and Impact on Project

- **Finding a place to meet in person**
- **Implementing the specs found from calculations into other parts of the system**

Expectations for Next Week

- **Complete week 9 progress report**
- **Prepare for presentation**
- **Continue working**

Name: Dae'Shaun Walden

Today's date: 10-23-20

Week Number: 9

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 9 progress report**
- **Prepare for presentation**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Complete week 9 progress report**
- **Completed and prepared for presentation**

Problems Encountered This Past Week and Impact on Project

- **Transistor PCB layout**
- **Gantt Chart adjustment**

Expectations for Next Week

- **Complete week 10 progress report**
- **Fix Gantt chart**
- **Finish transistor PCB layout**
- **Order all PCBs**
- **Get idea of physical robot design**

Name: Dae'Shaun Walden

Today's date: 10-30-20

Week Number: 10

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 10 progress report**
- **Finish transistor PCB layout**
- **Order PCBs**
- **Design robot design**
- **Make more progress on code**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Complete week 10 progress report**
- **Completed transistor PCB**
- **Power supply PCB ordered**
- **Made progress on the design of physical robot**
- **Made progress on code**

Problems Encountered This Past Week and Impact on Project

- **Transistor circuit terminal block**

Expectations for Next Week

- **Complete week 11 progress report**
- **Fix Gantt chart**
- **Continue robot design**
- **Report Progress**

Name: Dae'Shaun Walden

Today's date: 11-5-20

Week Number: 11

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 11 progress report**
- **Printed transistor PCB**
- **Solder power supply and transistor circuit PCBs**
- **Fix Gantt chart**
- **Establish Bluetooth connection on the android phone**
- **Create ideas and start building a physical robot**
- **Test movement of robot ideas**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 11 progress report**
- **Got the transistor PCB printed and soldered**
- **Soldered the power supply**
- **Fixed Gantt chart**
- **Establish and completed Bluetooth connection to the android phone**
- **Starting to build a physical robot**
- **Test proof of concept of robot ideas**
- **Completed presentation on progress**

Problems Encountered This Past Week and Impact on Project

- **Measurements for the terminal blocks of the transistor circuit**

Expectations for Next Week

- **Complete week 12 progress report**
- **Continue code**
- **Implement the transistor circuit with the power supply and microcontroller**

Name: Dae'Shaun Walden

Today's date: 11-13-20

Week Number: 12

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 12 progress report**
- **Test the transistor circuit board**
  - **Which will complete and confirm the electric side of the project**
- **Implement the transistor circuit with the power supply and microcontroller**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 12 progress report**
- **Tested the transistor circuit broad**
- **Completed the implementation test with the transistor circuit, power supply, and microcontroller**

Problems Encountered This Past Week and Impact on Project

- **Only one motor terminal block functions while the other terminal block for the second motor does not.**
    - **I believe this is a soldering issue.**

Expectations for Next Week

- **Complete week 13 progress report**
- **Complete project report draft**
- **Prepare and complete progress presentation**

Name: Dae'Shaun Walden

Today's date: 11-20-20

Week Number: 13

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 13 progress report**
- **Complete project report draft**
- **Prepare and complete progress presentation**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 13 progress report**
- **Prepared and complete the progress presentation**

Problems Encountered This Past Week and Impact on Project

- **The transistor motor control circuit is experiencing a huge voltage drop when it reaches the motor connection on the PCB board**
- **Constructing the robot on a new frame with new wheels and motors.**

Expectations for Next Week

- **Complete week 14 progress report**
- **Have transistor circuit fixed and implemented**
- **Complete project report draft**
- **Complete project poster**

Name: Dae'Shaun Walden

Today's date: 11-28-20

Week Number: 14

Total Hours for Week: 8

Expected Tasks to be Accomplished This Past Week

- **Complete week 14 progress report**
- **Have transistor circuit fixed and implemented**
- **Complete project report draft**
- **Complete project poster**

Actual Tasks Accomplished This Past Week and Number of Hours Spent on Each Task

- **Completed week 14 progress report**
- **Fixed and implemented the transistor circuit by changing the value of the resistor and changing the transistor for a transistor that can handle more current**

Problems Encountered This Past Week and Impact on Project

- **Completing on canvas assignments due to focusing on the main task which is implementing everything for the robot into one to prepare for the final presentation**

Expectations for Next Week

- **Prepare and turn in the final technical presentation to faculty**
- **Complete PEER assessment on Group Project**
- **Complete report draft**
- **Complete project poster final version**